VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

BAKALÁŘSKÁ PRÁCE

Brno, 2020 Kryštof Trávník

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ
DEPARTMENT OF TELECOMMUNICATIONS

ZÁTĚŽOVÝ ANALYZÁTOR PRO ENERGETICKÉ SÍTĚ
LOAD ANALYSER FOR INDUSTRIAL NETWORKS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **Kryštof Trávník**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. Tomáš Lieskovan**
SUPERVISOR

BRNO 2020

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní program **Informační bezpečnost**

Ústav telekomunikací

Student: Kryštof Trávník ***ID:*** 193956
Ročník: 3 ***Akademický rok:*** 2019/20

NÁZEV TÉMATU:

Zátěžový analyzátor pro energetické sítě

POKYNY PRO VYPRACOVÁNÍ:

Cílem práce je vytvořit analyzátor schopný analyzovat výkon (zpracování požadavků za sekundu) jednotlivých zařízení v energetice (smart metrů, koncentrátorů, bran, modemů) při paralelním zasílání útoků na odepření služeb (DoS). Samotný program bude zpracován v jazyku Java nebo Python a výsledky budou prezentovány pomocí webové aplikace.

DOPORUČENÁ LITERATURA:

[1] Gurux DLMS Server: https://www.gurux.fi/Gurux.DLMS.Server

[2] JIRKA, Bc Matěj. Framework DLMS/COSEM pro sběr dat v AMM systémech.

Termín zadání: 3.2.2020 ***Termín odevzdání:*** 8.6.2020

Vedoucí práce: Ing. Tomáš Lieskovan

doc. Ing. Jan Hajný, Ph.D.
předseda rady studijního programu

ABSTRAKT

Bakalářská práce se zabývá vytvořením aplikace, složené z webové aplikace a webového serveru. V teoretické části je rozebrán protokol DLMS a specifikace COSEM, útoky na odepření služeb, nástroje, které útoky dokáží zrealizovat, a nakonec rozebrány klíčové komponenty Dockeru. Vytvořená aplikace odesílá na metr paralelně požadavek na objekt metru a útok na odepření služeb. Ke generování DoS útoků je využíván nástroj Hping3 a je možné vybrat mezi více typy útoků. Požadavky na metr jsou odesílány a přijímány pomocí aplikace, vytvořené v jazyce Java. Výsledky měření jsou následně prezentovány ve webové aplikaci, a to formou grafu a naměřených dat ve formátu CSV.

KLÍČOVÁ SLOVA

DLMS, COSEM, Hping3, DoS, Raspberry Pi, Docker, Java, Energetika, Elektroměr

ABSTRACT

The bachelor thesis is focused on developing an application, which is consisted of a web application and a web server. In the theoretical part are described topics such as DLMS protocol, specificationCOSEM, attacks on service denial, tools for generating denial attacks, and key parts of Docker. The app which is created is sending a request to smart meters together with Denial of service attacks. Hping3 is used for generating DoS attacks and several options of attack are available. The app is created in Java. Received results are shown in form CSV and also graphically in the web application.

KEYWORDS

DLMS, COSEM, Hping3, DoS, Raspberry Pi, Docker, Java, Energetics, Smartmeter

TRÁVNÍK, Kryštof. *Zátěžový analyzátor pro energetické sítě*. Brno, 2020, 59 s. Bakalářská práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedoucí práce: Ing. Tomáš Lieskovan.

PROHLÁŠENÍ

Prohlašuji, že svou bakalářskou práci na téma „Zátěžový analyzátor pro energetické sítě“ jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autora

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu bakalářské práce panu Ing.Tomáši Lieskovanovi za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Obsah

Seznam obrázků

Seznam výpisů

Úvod

Bez elektrické energie, potažmo energetiky, si již v dnešní době téměř nelze život představit. Zabezpečení nepřerušené dodávky energie i v krizových situacích[1] v rozsahu nezbytném pro fungování státu je rovněž součástí Státní energetické koncepce [2]. Odvětví energetiky je tímto součástí kritické infrastruktury [3] a to dokonce jako jedno z nejvýznamnějších. Kritickou infrastrukturou se rozumí, že narušení jakéhokoli jejího prvku by mohlo mít katastrofální dopad na bezpečnost státu, zabezpečení základních životních potřeb obyvatelstva, zdraví osob nebo ekonomiku státu. Prvkem kritické infrastruktury se pak rozumí stavba, zařízení, prostředek nebo veřejná infrastruktura. V kontextu této práce se budeme zabývat konkrétně elektroměrem. Dalším podstatným pojmem je subjekt kritické infrastruktury [4] a tím se rozumí provozovatel prvku kritické infrastruktury. Jednou z povinností subjektu je pravidelná kontrola a testovaní své infrastruktury. Do této povinnosti spadá i testovaní proti kybernetickým útokům, které jsou v dnešní době neustále na vzestupu a je třeba hledat nástroje pro jejich identifikaci a následné efektivní zabezpečení. Z těchto důvodů je nezbytné nepřetržité analyzování zařízení pomocí testů a přidávání dalších prvků zabezpečení. V rámci této práce jde o vytvoření návrhu analyzátoru, který je schopen analyzovat výkon elektroměru při paralelním zasílání útoků na odepření služeb.

Popisem komunikace mezi zařízeními se zabývají komunikační protokoly. Definují způsob, jakým probíhá komunikace na všech úrovních. Protokol je tak známý oběma komunikujícím stranám a jsou nedílnou součástí každé počítačové komunikace. Bez takových protokolů bychom si jen těžko mohli představit funkce jako detekce okolních zařízení či zahájení a ukončení komunikace. V dnešním době existuje obrovské množství protokolů, které mohou plnit funkce základní. Pak ale existují i protokoly, které jsou mnohem více komplexní. Mezi takové protokoly patří mimo jiné průmyslové protokoly, jejichž hlavním úkolem je vytvořit a automatizovat některé průmyslové operace. Mezi nejznámější a nejvíce rozšířené protokoly patří Modbus (Modicon Communication Bus) a DNP3 (Distributed Network Protocol). Do budoucna se však nejvíce počítá s protokolem DLMS (Device Language Message Specification), který má být v budoucnu používán jako standard. Protokol je postaven na rozhraní tříd COSEM (Companion Specification for Energy Metering), který obsahuje specifikace, jež definují transportní a aplikační vrstvu DLMS protokolu.

Tato semestrální práce se nejdříve zabývá studií specifikací DLMS/COSEM, následně seznámení s nástroji, které jsou schopny útoky simulovat a nakonec analýzou a popisem chování zařízení při zasílání požadavku souběžně s útokem na odepření služeb. Jedná se o malý audit prvků v laboratorním pracovišti.

1 Průmyslové komunikační sítě

Průmyslové sítě jsou obecně určeny pro přenos dat ve velkém měřítku. Propojují různá zařízení napříč vzdálenými prostory a díky tomu jsme schopni si mezi sebou přeposílat data. Průmyslové sítě jsou navrženy tak, aby v reálném čase splňovaly potřeby a požadavky velkého počtu systémů.

1.1 Rozmach průmyslových sítí

Dle knihy [5] se s příchodem internetu v energetice používaly převážně kabelové systémy, v praxi to znamenalo, že každý přepínač nebo relé bylo připojeno k programovatelnému logickému automatu (PLC) pomocí kabelu, což přinášelo problém s obrovským množstvím kabelů. Proto postupně došlo k jejich nahrazení průmyslovými sítěmi. Dnes už se však staly standardem a jen těžko budeme hledat vetší společnost, která sítě nevyužívá. Jeden z hlavních důvodů, proč jsou dnes průmyslové sítě tak rozšířené, je časové hledisko. Už při výrobě jednotlivých komponentů zařízení se hlavně díky nepotřebným kabelům zkracuje doba výroby, tím pádem se k zákazníkovi dostanou mnohonásobně rychleji a ze stejného důvodu se také ušetří čas při instalaci těchto zařízení, což je v konečném důsledku výhodné pro obě strany.

Další významný důvod je možnost řešit problémy, které nastanou, přes síť, bez nutnosti fyzického přístupu k danému zařízení. Proč však vlastně vůbec průmyslové sítě či protokoly existují, když bychom mohli používat stejné protokoly i zařízení jako v domácích (pracovních) sítích? Na průmyslové sítě máme jednoduše jiné požadavky, například restartovat router v naší domácí síti není vůbec žádný problém. Ovšem restartovat zařízení uprostřed pouště, vzdálené několik stovek kilometrů od nejbližšího pracoviště už tak jednoduché nebude. To stejné platí pro klimatické podmínky. Zatímco zařízení, které je určené do domácího prostředí není vybaveno ochranou pro extrémní klimatické podmínky. V průmyslových sítích se s takovou ochranou počítat musí.

1.2 Oddělení sítí

Kniha [5] dále uvádí, že z bezpečnostních důvodů a také z důvodu lepšího výkonu na úrovni průmyslových sítí je doporučeno izolovat pracovní síť od průmyslové. Propojení mezi pracovní a průmyslovou sítí musí být přes bránu firewall, aby se omezil přístup a zajistila se bezpečnost od venkovních kybernetických útoků.

Průmyslové sítě jsou dále rozděleny do několika podsítí a to hlavně z důvodu snížení nežádoucího síťového provozu a zvýšení rychlosti odezvy sítě. Snížení síťového provozu v průmyslových sítích je opravdu důležité a abychom toho dosáhli, tak

většina výrobců DCS a PLC používá hlášení procesních parametrů založených na výjimkách. DCS/PLC zapisuje do libovolného digitálního bodu (zapnuto nebo vypnuto) jen když dojde ke změně hodnoty z vypnuto na zapnuto nebo naopak, nikoliv v pravidelné frekvenci a tím se sníží zatížení sítě. Dalším důležitým prvkem průmyslových sítí jsou přepínače, což jsou inteligentní zařízení druhé vrstvy, díky kterým se snižuje pravděpodobnost kolize a zvyšuje celková rychlost přenosu. Přestože se doporučuje izolovat pracovní síť od průmyslové sítě i tato integrace přináší několik výhod. Obecně se doporučuje norma ISA 95. Tato norma doporučuje různé úrovně průmyslové sítě a jak díky ním dosáhnout lepší integrace pracovní a průmyslové sítě a zároveň neohrozit bezpečnost. Následující úrovně jsou doporučeny ISA 95.

- Level 0: definuje skutečné fyzické procesy nebo fyzické výrobní procesy.
- Level 1: definuje činnosti spojené se snímáním a manipulací s fyzickým procesem a automatické řízení parametrů procesu. To zahrnuje senzory, vstupní a výstupní (I/O) moduly a vestavěné ovladače.
- Level 2: definuje monitorování a kontrolu fyzických procesů. Rozhraní člověk – stroj, SCADA a další jsou součástí druhé úrovně.
- Level 3: definuje aktivity pracovního postupu. Na této úrovni se nachází údržba informací, která je centralizovaná a poskytuje lepší kontrolu a dostupnost záznamů.
- Level 4: definuje obchodní činnosti potřebné k řízení výrobní organizace. Plánovaní podnikových zdrojů je kritickou součástí, která se nachází na této úrovni.

2 Základní popis DLMS/COSEM

Kapitola je inspirována ze specifikace Green book [6] a z bakalářské práce [7]. Device Language Message Specification a Companion Specification for Energy Metering, zkráceně DLMS/COSEM. Za vytvořením specifikace stojí firma DLMS UA. Specifikace byla rozdělena do čtyř knih, které jsou od sebe odlišeny pomocí barev. Soubor těchto knih se nazývá Coloured Books a každá z knih tvoří jedinečnou dokumentaci popisující jinou část a dohromady tvoří celek. Kniha Green se zabývá architekturou a protokoly, kniha Blue[8] pak popisuje COSEM objekty a OBIS[1], kniha Yellow [9] se zabývá testováním a kniha White je slovník pojmů.
DLMS/COSEM specifikace se skládá z tří následujících kroků:

- krok 1 – **modelování.**
  Zahrnuje model rozhraní měřícího zařízení a pravidla pro identifikaci dat, je zahrnuto v knize Blue,
- krok 2 – **zasílání zpráv.**
  zahrnuje služby na mapovaní modelu rozhraní do datové jednotky (APDU) a také kódování těchto APDU,
- krok 3 – **přenos.**
  zahrnuje přenos zpráv prostřednictvím komunikačního kanálu, společně s krokem dva je zahrnut v knize Green.

Aplikační vrstva DLMS/COSEM popisuje služby, které slouží k navázaní logického spojení mezi klientem a serverem. Dále definuje služby, které slouží k přístupu do COSEM objektů a jejich atributům a metodám. Profily specifické pro komunikační média DLMS/COSEM určují, jak lze zprávy aplikační vrstvy přenášet přes různá komunikační média. Každý komunikační profil určuje sadu vrstev protokolů, požadovaných pro podporu horní aplikační vrstvy DLMS/COSEM. Jak už je známo, v poslední době se rozšiřuje používaní inteligentních měřících systémů, a proto jsou vyžadovány silné mechanismy pro zabezpečení informací, které ochrání soukromí odběratelů energií, obchodní zájmy poskytovatelů energií a v neposlední řadě musí být zabezpečena celá energetická infrastruktura. DLMS/COSEM obsahuje vestavěné bezpečností mechanismy již od svého vzniku, konkrétně se jednalo o mechanismy pro identifikaci a autentizaci klientů a serverů. Všechno začíná už od samotného spojení, které se nazývá „Application Associations“. Přenášené zprávy APDU (Application Protocol Data Unit) mohou být šifrované, aby si klient a server mohl zprávy bezpečně vyměnit. S rostoucí bezpečnostní hrozbou byly přidávány další elementy ochrany, mezi nejdůležitější patří, že ochrana může být použita nejen mezi serverem a klientem, ale také mezi aplikacemi třetích stran[2]. K dispozici jsou symetrické al-

[1] OBject Identification System, více v kapitole 2.11
[2] Aplikace, které nevyužívají DLMS/COSEM

goritmy a algoritmy veřejných klíčů, které je možné libovolně kombinovat a to pro autentizaci, šifrovaní anebo digitální podpis.

2.1 Výhody DLMS/COSEM

Uvádí se [10], že nejvíce obecně používaným standardem pro výměnu dat je FLAG, který je pro účel energetické měřící techniky definován v IEC 61107 podle IEC TC 13. Další protokoly, které stojí za zmínku jsou:

- Euridus, který je používaný především ve Francii přes kroucenou dvojlinku, standardizovaný v IEC 62056-31:1999 podle IEC TC 13 pro měření elektřiny,
- MBUS pro měření ohřívání, standardizován v EN 1434-3:1997 podle CEN TC 294,
- dále pak IEC 60870-5-102:1996 pro přenos mezi přenosovými a distribučními stanicemi, standardizován podle IEC TC 57.
- V severní Americe se pak používá ANSI C12.18, C12.19 a C12.21.

Cílem zde není podrobně srovnávat výše zmíněné protokoly, ale poukázat na prvky, které dělají z DLMS/COSEM vhodný protokol pro energetické odvětví. Rozhraní DLMS/COSEM definuje model, který se dá uplatnit na všechny obory energetiky jako je elektrárenství, plynárenství, vodárenství a tak dále. Každý objekt v rozhraní má vlastní unikátní identifikátor, který data rozpozná a pošle přes komunikační linku. Model je pak zcela nezávislý na vrstvách protokolů přenášejících data. Podporuje inovaci a budoucí rozvoj tím, že výrobcům umožňuje přidávat vlastní objekty, atributy, metody a možnost vytvářet úplně nová rozhraní a verze bez nutnosti měnit přístup k objektům a tím udržovat interoperabilitu. Rozhraní tříd standardizuje velké množství funkcí měřiče, jako je například měření kvality energie, synchronizace časů při výpadku napájení a vytváření tarifů. Jednoznačná interpretace dat je zajištěna definováním datových typů, které mají být využity pro atributy a také nutnost zaslat informace zároveň s daty. DLMS/COSEM poskytuje kontrolovaný a bezpečný přístup k informacím o jednotlivých smart metrech. Jelikož je model rozhraní naprosto nezávislý na komunikačním médiu, nemusíme měnit ani model, ani aplikaci pro správu dat. DLMS/COSEM usnadňuje výrobu ovladačů, protože mohou být používány na různé typy měřičů a i na měřiče od odlišných výrobců. Byl vybrán komunitou pro měření elektřiny, vody a plynu jako standard, se kterým se do budoucna počítá a to je výhodné pro všechny výrobce a poskytovatele, především pak pro ty začínající, kteří se soustředí na větší škálu energetického odvětví.

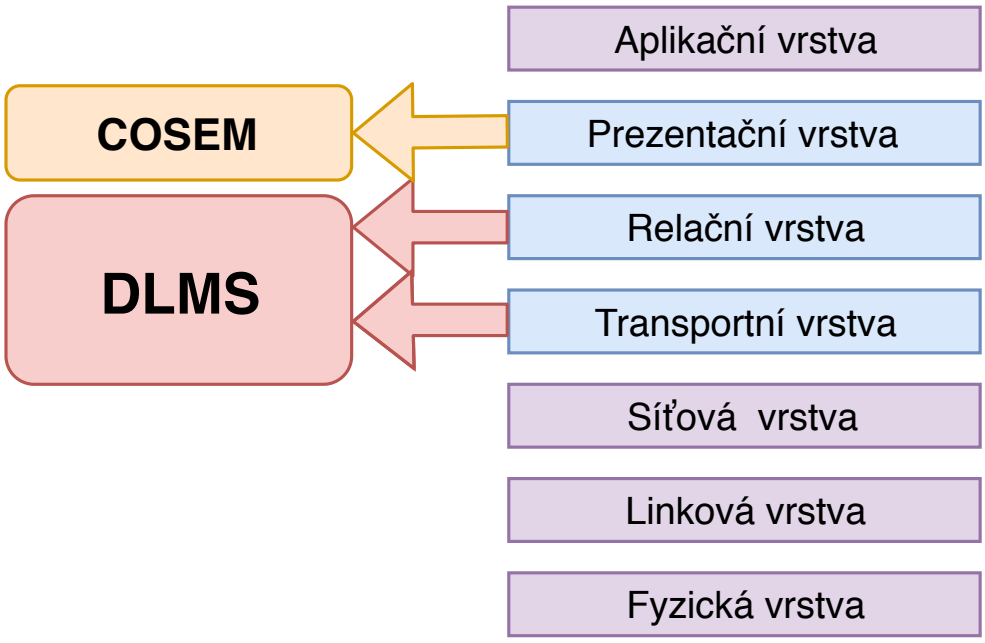
Jedinečná kombinace funkcí, které nejsou dostupné u žádného jiného protokolu, který je v současnosti známý, nebo používaný, dělají z DLMS/COSEM atraktivní protokol pro všechny, kdo chtějí výrazně snížit svoje náklady a následně se rozvíjet.

2.2 Device Language Message Specification

Device Language Message Specification (DLMS) je specifikace aplikační vrstvy navržená pro podporu zasílaní zpráv do a z energetických zařízení. Podporovány jsou aplikace pro dálkové odečty měřičů, dálkové ovládaní a služby pro měření jakékoli energie, jako elektřina, voda, plyn a teplo. DLMS je vyvinutý a udržovaný společnosti DLMS User Asscociation (DLMS UA). Dále protokol obstarává přenos informací do nižších vrstev. Vrstvy se starají o samotný přenos dat.

2.3 DLMS/COSEM na modelu OSI/ISO

Open Systems Interconnection/International Standards Organization (OSI/ISO) je komunikační model, jež definuje komunikaci mezi dvěma zařízeními v síti [11]. Skládá se ze sedmi vrstev, nás bude nejvíce zajímat vrstva prezentační, relační a transportní. Prezentační vrstva má na starost syntaxi přenášených dat mezi počítači, podobnou funkci pak obstarává objektový model COSEM, který data taktéž formátuje. Relační vrstva má na starost vytváření a správu relací mezi dvěma počítači, transportní vrstva pak obstarává rozdělení dat z relační vrstvy na menší pakety a zajišťuje, že pakety dorazí na druhý konec. Podobné úkoly plní DLMS, konkretně dohlíží na dialog a výměnu dat. Názorně tedy můžeme umístit DLMS a COSEM na komunikační model OSI/ISO viz obrázek 2.1.



Obr. 2.1: Znázornění DLMS a COSEM v komunikačním modelu OSI/ISO.

2.4 Jména

Všechny subjekty DLMS/COSEM, včetně klientů, serverů a systémů třetích stran, musí mít unikátní identifikátor, pojmenovaný podle názvu jejich systému neboli systémové jméno a tento identifikátor jim musí být trvale přidělen. Fyzická zařízení serveru mohou hostit jedno nebo více logických zařízení. Logická zařízení následně musí být unikátně pojmenována podle jejich „Logical Device Name“. Logická zařízení, která sdílejí stejné fyzické zařízení pak sdílejí stejné systémové jméno, které je více popsáno v kapitole 2.6.

2.5 Adresace

Každé fyzické zařízení musí mít odpovídající adresu. Adresa závisí na komunikačním profilu a může to být telefonní číslo, MAC adresa, IP adresa nebo jejich kombinace. Adresy mohou být nakonfigurovány předem a nebo mohou být přiděleny během registračního procesu, který je svázán se systémovým jménem, které je popsáno v kapitole 2.6. Všechny servery a klienti, kteří jsou součástí DLMS/COSEM, jsou vázáni na Service Access Point (SAP). SAP se nachází v podpůrné vrstvě DLMS/COSEM AL. V závislosti na komunikačním profilu může SAP zastupovat například TCP-UDP/IP wrapper. Na straně serveru je tato vazba modelována pomocí SAP Assignment. Více podrobností se nachází v Blue book [8].

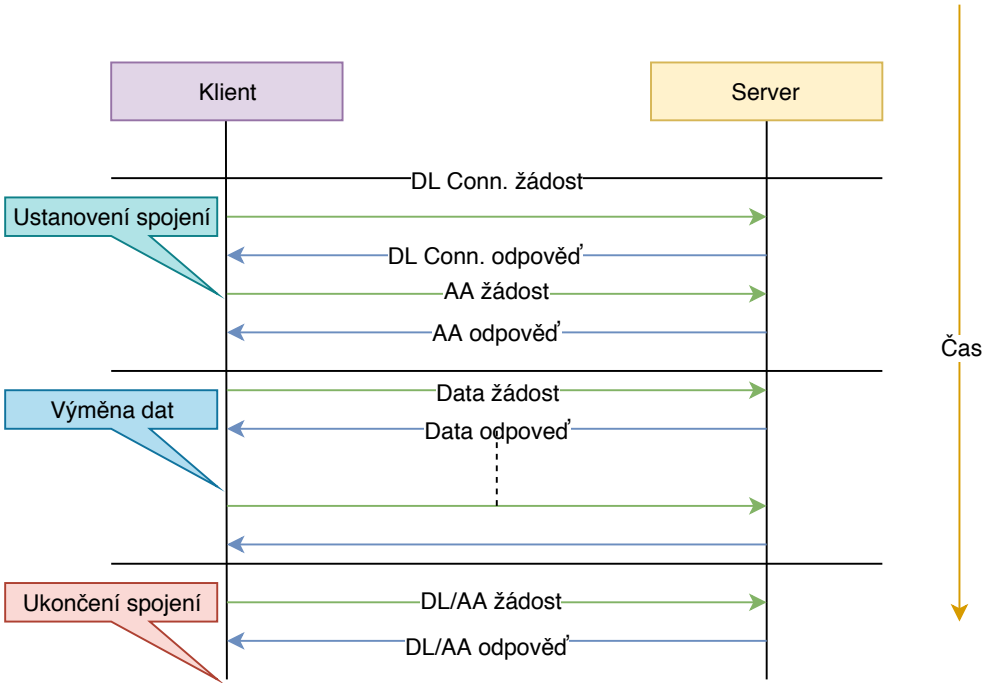
2.6 Systémové jméno

Systémové jméno, též známo pod značením Sys-T, jednoznačně identifikuje každou entitu v DLMS/COSEM, což může být klient, server nebo třetí strana a může přistupovat k serverům prostřednictvím klientů. Systémové jméno (Sys-T) musí splňovat dvě podmínky, unikátnost a délku 8 oktetů. První tři oktety obsahují ID výrobce a dalších pět pak musí být naprosto unikátní. ID výrobce přiděluje DLMS a FLAG Association [12]. Na oficiálních stránkách DLMS se uvádí, že je registrováno 1303 ID flagů, ale pří procházení databáze jich najdeme pouze 1192. V případě broadcastové komunikace posílá systémové jméno jen klient serveru. Systémové jméno se dále používá k zabezpečení zpráv a dat v objektech COSEM pomocí kryptografických algoritmů. Předtím než se však aplikují, strany si mezi sebou musí vyměnit systémové jméno, které může být vyměněno během Application Association spojení, pomocí COSEM-OPEN služby. Pokud pak Application Association obdržená během tohoto spojení nejsou stejná jako Application Association obdržena během registračního procesu, musí být zamítnuta.

2.7 Connection oriented operation

Komunikační spojení se skládá ze tří kroků, viz obrázek 2.2.

- Nejdříve je navázáno spojení na aplikační vrstvě, které nese název „Application Association“ (dále jen AA), mezi klientem a serverem. Každá vrstva může mít navázáno jedno nebo více spojení současně.
- Jakmile je spojení ustaveno, nastává výměna dat.
- Po výměně dat je AA spojení ukončeno.



Obr. 2.2: Přehled DLMS komunikace.

Pro účely jednoduchých zařízení je povolena komunikace jednosměrná, multicastová[3] a broadcastová[4]. Využívá se vytvořeného spojení AA, u kterého se předpokládá, že bylo vytvořeno v minulosti. Tím pádem se mohou data začít rovnou vyměňovat.

2.8 Application associations

AA jsou logická spojení mezi klientem a serverem. AA spojení se s serverem naváže na žádost klienta nebo se již pokládá za vytvořené v minulosti. Žádosti mohou být potvrzeny nebo zamítnuty. Servery nemohou inicializovat AA spojení. Každé zařízení COSEM může navázat jedno nebo více AA s různými klienty. Spojení AA může být potvrzené nebo nepotvrzené. **V potvrzených AA**:

[3]Zpráva v počítačové síti odeslaná z jednoho zdroje, kterou příjme skupina více zařízení.
[4]Zpráva v počítačové síti, kterou přijmou všechna zařízení.

- klient může odesílat potvrzené servisní požadavky a server na ně odpovídá – pull operace.
- Klient může poslat nepotvrzené servisní požadavky a server neodpovídá.
- Server může posílat klientovi nevyžádané žádosti, které má nastavené a provádí se automaticky nebo pokud se stane určitá událost – push operace.

V nepotvrzených žádostech AA:

- server nesmí reagovat a nemůže iniciovat servisní požadavky – to může pouze klient.

2.9 Autentizace v DLMS

V komunikačních systémech je autentizace zařízení stěžejní funkcí a velmi důležitou bezpečnostní službou. Cílem autentizace je zjistit, jestli je žadatel určité služby skutečně ten, kdo o službu žádá. Abychom dosáhli tohoto cíle, měl by existovat již vytvořený vztah, který by měl spojovat zařízení se službou. V DLMS/COSEM probíhá autentizace během spojení AA. V potvrzených AA se může autentizovat buď klient (jednostranné ověření), nebo klient i server (vzájemné ověření). V nepotvrzené AA se může autentizovat pouze klient. V minulosti vytvořených AA není autentizace k dispozici. Jakmile je autentizace úspěšná, může ke COSEM objektům a metodám klient přistupovat.

Dále DLMS rozlišuje tři druhy autentizace podle úrovně zabezpečeni.

1. **Žádné zabezpečení (nejmenší úroveň ochrany)** – umožní klientovi získat některé základní informace ze serveru. Ověření nevyžaduje žádnou autentizaci a klient může přistupovat k atributům a metodám objektu COSEM v kontextu přístupových práv uvedených v dané AA.
2. **Nízká úroveň zabezpečení (LLS)** – server vyžaduje od klienta heslo, které zná a tím se autentizuje. Heslo je drženo aktuálním „Association SN[5]/LN[6]“ objektem. Pokud je heslo přijato, je navázáno spojení AA, v opačném případě je spojení zamítnuto a přerušeno. Ověřování je v rámci žádosti služby COSEM-OPEN.
3. **Vysoká úroveň zabezpečení**– server i klient se musí sami vzájemně autentizovat, aby došlo k navázání AA. Autentizace na nejvyšší úrovni je čtyřkrokový proces, který je opět v rámci služby COSEM-OPEN. V prvním kroku klient přenáší žádost „challenge klient to server“ a v závislosti na zvoleném algoritmu poskytuje serveru další informace. Ve druhém kroku přenáší server žádost „challenge server to klient“ a opět informace závislé na zvoleném algoritmu. Pokud je „challenge klient to server“ stejná jako „challenge server to

[5]Třída, která pomáhá připojenému zařízení k získání všech SN daných objektů.

[6]Třída pro asociaci navázanou pro referencování pomocí LN.

klient“, klient zamítne a zruší pokus o spojení. Pokud není stejná, tak klient ověří „challenge server to klient“ a další informace podle algoritmu, uzná výsledek jako platný a odešle ho serveru. Server zkontroluje, jestli je „challenge server to klient“ výsledek správný a pokud ano, schválí autentizaci klienta. V posledním kroku pak server zpracuje „challenge klient to server“ a další informace podle algoritmu a výsledek odešle klientovi. Klient zkontroluje, jestli je „challenge klient to server“ výsledek správný a pokud ano, schválí autentizaci serveru.

Algoritmy, které jsou k dispozici jsou: MD5, SHA-1, SHA-256, EC-DSA a GMAC.

2.10 Companion Specification for Energy Metering

Companion Specification for Energy Metering neboli COSEM je model rozhraní energetických měřících zařízení, který poskytuje funkcionalitu dostupnou skrze komunikační rozhraní. Používá se objektově orientovaný přístup.

COSEM modeluje fyzické měřící zařízení jako sadu logických zařízení. Každé logické zařízení má celosvětový jedinečný identifikátor, který se nazývá „logical device name“. Informace, které jsou obsaženy v každém logickém zařízení jsou modelovaný jako objekty rozhraní. Přistup k objektům rozhraní v rámci logického zařízení je modelován asociačními objekty. Asociační objekt poskytuje informace o prostředcích dostupných v logickém zařízení, v závislosti na přístupových právech.

Objekty rozhraní jsou specifické pro měřící doménu a využívají dobře známe koncepty jako registr, registr požadavků, profily, hodiny a další. Existují také speciální objekty, které řídí přístup a konfigurují komunikační kanály.

Informace uchovávané v objektu rozhraní, jsou uspořádány do atributů. Atributy reprezentují vlastnosti objektu. Objekt může disponovat řadou metod, které buď atributy čtou, nebo je mohou libovolně upravovat. Objekty, které sdílí stejnou charakteristiku (stejné atributy a metody), tvoří třídu rozhraní, která se identifikuje verzí a ID třídy.

Prvním atributem jakéhokoli objektu je logické jméno, které je součástí identifikátoru objektu. Logické jméno, ID třídy a verze jednoznačně identifikuje význam informací, které jsou v jakémkoli objektu. Dále model COSEM umožňuje identifikaci a čtení informací, které jsou uchovávány v jakémkoli měřidle a to nezávisle na výrobci tohoto měřícího zařízení. Vše se odehrává zabezpečeným způsobem.

2.11 Object Identification System

Object Identification System neboli OBIS je popsán v literatuře [8] a definuje identifikační kódy pro běžně používané datové systémy v měřícím průmyslu.

OBIS poskytuje unikátní identifikátor pro měřící zařízení a vztahuje se nejen na naměřené hodnoty, ale také na abstraktní hodnoty, které jsou používány na konfiguraci nebo získávání informací o chování těchto zařízení. ID kódy jsou použity na identifikaci:

- Logických jmen a objektů,
- dat přenesených přes komunikační kanály,
- dat, která se zobrazují na měřicích zařízeních.

OBIS kód je kombinace šesti různých skupin, které hierarchicky popisují význam každé z nich. Každá skupina nabývá číselnou hodnotu. Příklad OBIS skupin, které pak tvoří celek, lze vidět na tabulce 2.1.

A	B	C	D	E	F

Tab. 2.1: Struktura OBIS kódu.

Popis jednotlivých skupin:

- skupina A – definuje charakteristiku dat, která budou požadována (elektřinu, plyn a další),
- skupina B – definuje kanál, v případě, že jsou zasílána stejná nebo podobná data z více zdrojů a to umožňuje data rozlišit,
- skupina C – popisuje přesněji charakteristiku dat a váže se ke skupině A. Pokud je skupina A elektřina, skupina C pak bude například napětí nebo proud,
- skupina D – více upřesňuje výsledek na základě hodnoty skupin A a C, podle různých specifických algoritmů,
- skupina E – podrobněji zpracovává výsledky z A až D do registrů,
- skupina F – definuje úložiště dat, pro různá období.

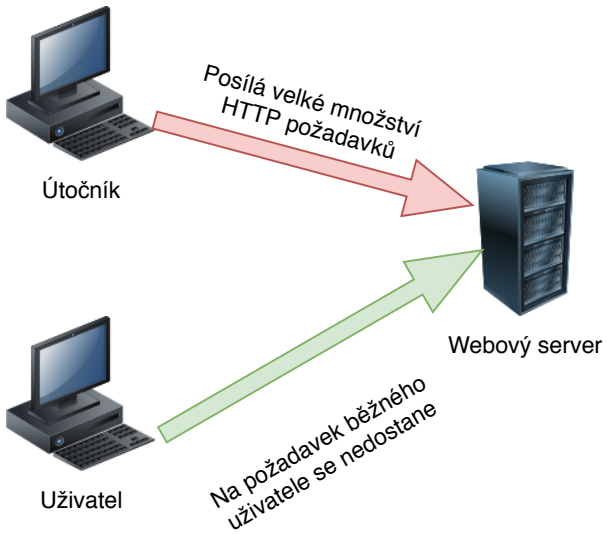
Pokud není některá ze skupin relevantní pro daný úkon, využívá se pro bližší klasifikaci dat. Podrobnější popis OBIS skupin lze nalézt v literatuře [13]

3 Nástroje na vytížení linky

V následující kapitole budou přiblíženy nástroje, kterými bude testováno a útočeno na měřící zařízení a poté bude popisováno jeho chovaní. Útoky, které budou v této kapitole popsány, jsou často známy i mezi širokou veřejností a to převážné díky útokům na nadnárodní korporace a velké firmy jako Microsoft či Yahoo.

3.1 Útok na odepření služeb

Autoři útok Denial of service, více znám pod zkratkou DoS a nebo „záplavový útok“, popisují v článku [14]. Hlavní úkol DoS útoku spočívá v zahlcení cílové služby, sítě nebo zařízení zprávami takovým způsobem, že služba nebude schopna odpovídat na dotazy běžným uživatelům. Příklad primitivního HTTP DoS útoku vidíme na obrázku 3.1. Nemožnosti odpovídat na dotazy běžným uživatelům se docílí bombardováním cíle automaticky generovanými žádostmi.



Obr. 3.1: Útok typu DoS.

Ping flood, znám také pod pojmem ICMP flood [15], je útok DoS, ve kterém útočník shodí cílový počítač nebo zařízení tím, že jej přehltí požadavky typu ICMP echo, také známými jako ping. Útok zahrnuje „zaplavení“ sítě oběti pakety typu požadavek, protože je známo, že síť bude reagovat stejným počtem paketů s odpovědí. Dalšími metodami, jak zahltit cíl, mohou být vlastní implementace nebo využití programů jako jsou Hping, Scapy a mnoho dalších. V každém případě se tak vyčerpává šířka pásma a to vede k odmítnutí služby.

Za normálních okolností se ping používá k testovaní konektivity dvou počítačů měřením doby, za jakou urazí ICMP požadavek cestu k vzdálenému počítači a doby

za jakou je ICMP odpověď obdržena zpátky na prvním zařízení. Během útoku se však ping používá k přetížení cílové sítě datovými pakety. U útoku ping flood je důležitá znalost, jakou útočník o cíli disponuje. Útok rozdělujeme na tři kategorie, které se odvíjí od znalosti cíle:

1. **Útok na jeden počítač v lokální síti**. Útočník musí mít znalost IP adresy tohoto zařízení. Úspěšný útok by vedl k nemožnosti používaní tohoto zařízení v síti.
2. **Útok mířený na směrovač v síti**. Cílem je narušit komunikaci v síti. Útočník musí disponovat veřejnou adresou směrovače. Útok je považován za úspěšný, pokud zařízení v síti nebudou schopny komunikovat se směrovačem.
3. **Útok na bez znalostí**. Pokud útočník nedisponuje žádnou z těchto znalostí, musí prvně IP adresu získat pomocí programu třetí strany a poté pokračovat krokem jedna nebo dva.

Důležité je poznamenat, že pokud by měl být útok úspěšný, muselo by mít útočící zařízení mnohem větší šířku pásma než oběť. Proto se útok používá převážně s pomocí botů ve verzi DDoS (viz níže).

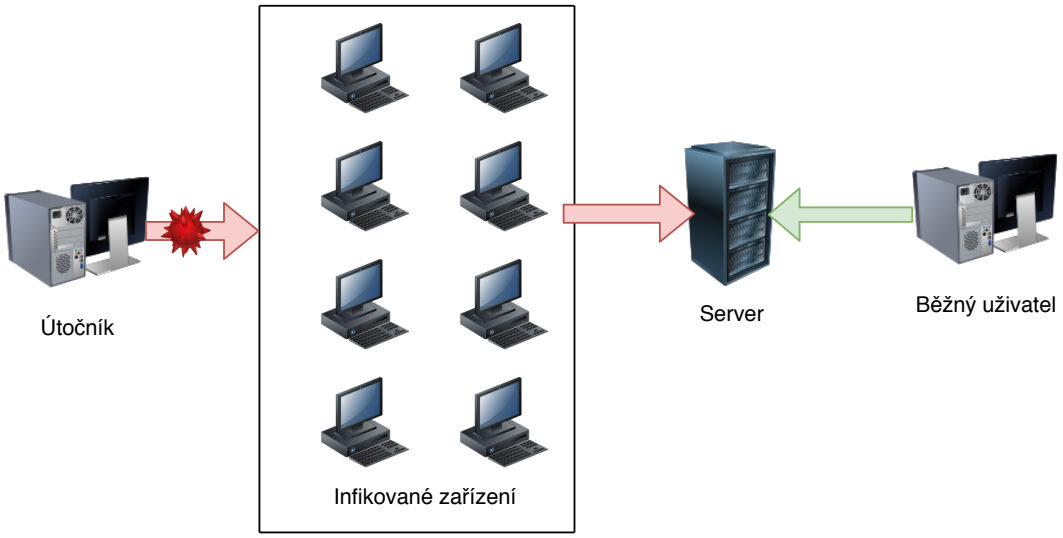
3.2 Distribuovaný útok na odepření služeb

Distrubuted Denial of Service (DDoS) je podtypem DOS a je v současnosti nejčastěji používaným útokem. Jeho popis je v článku [16]. Místo jednoho zařízení útočí mnohem více takových zařízení, viz obrázek 3.2. Zpravidla se však zařízení stala terčem škodlivého softwaru, jako je třeba trojský kůň, [1] a proto jejich majitelé často neví, že se útoku účastní. Například se může jednat o malware, který má už dopředu staticky nastavenou IP adresu oběti a čas útoku aktivace. Z hlediska útočníka má distribuovaný DoS několik výhod. Hlavní výhodou je větší množství zařízení, tudíž je možné generovat větší množství síťového provozu a zatížit tak cíl útoku ještě víc. DDoS útoky pak můžeme rozdělit do tří kategorií a to útoky na aplikační vrstvu, na protokoly a na záplavové útoky [17].

3.2.1 Útoky na aplikační vrstvu

Útoky jsou soustředěny na 7. vrstvu modelu OSI. V této vrstvě jsou vytvářeny webové stránky na žádost koncového uživatele. Žádost není pro klienta velkou zátěží a může jich tak generovat i více. Ovšem odpověď serveru na klientskou žádost již server zatěžuje, jelikož musí podle požadavku sestavovat všechny stránky, vypočítat dotazy a nebo načítat výsledky z databáze. Jedná se o útoky typu HTTP Flood, které ve verzi DoS můžeme vidět na obrázku 3.1 a také o útoky na služby DNS.

[1] Trojský kůň je uživateli skrytá část programu nebo aplikace



Obr. 3.2: Útok typu DDoS.

V záplavovém útoku HTTP [18] útočník posílá velké množství legitimních HTTP GET nebo HTTP POST požadavků na webový server nebo aplikaci. Sofistikovanější útoky nevyužívají poškozené pakety a spoofing a požadují menší šířku pásma než ostatní útoky, které mají za cíl poškodit nebo úplně shodit cílovou stránku nebo server. Předpokládá to ale mnohem rozsáhlejší a detailnější informace o cílové stránce nebo aplikaci. V takovém případě je pak útok ušitý na míru určitému cíli a je mnohem těžší takový útok detekovat. Když HTTP klient, což může být například webový prohlížeč, dotazuje server, zasílá HTTP žádost – GET nebo POST. GET je využívána při načítání statického obsahu jako obrázky, POST se používá pro přístup k dynamicky generovaného obsahu, jako je třeba formulář. Útok je nejvíce efektivní pokud server nebo aplikace přidělí maximální možné zdroje každému požadavku. Z toho důvodu je používání POST žádostí více účinné, protože zahrnují parametry, které spouštějí komplexnější procesy na straně serveru. Na druhou stranu GET žádosti jsou jednodušší na vytvoření. Záplavové útoky HTTP je těžké odlišit od normálního provozu, protože používají standardní požadavky URL. Detekce založená na sledování četnosti sledovaného jevu je neúčinná, protože při detekovaní HTTP útoku je objem provozu pod stanovenými detekčními prahy.

3.2.2 Útoky na protokoly

Rovněž známý pod anglickým názvem „state–exehaustion attacks“. Zde se útok se zaměřuje na slabiny třetí a čtvrté vrstvy OSI modelu. Jedná se o vrstvu síťovou a transportní. Úkolem útoku je vyčerpat cílové zdroje jako porty a firewally. Jedná se

o záplavový útok na SYN[2] a o útok zvaný „ping of death“, který využívá slabiny v protokolu TCP/IP.

Záplavový útok SYN [19] je typ útoku, který využívá slabinu takzvaného „TCP tree–way handshaku“ tím, že vyčerpá všechny volné prostředky cílového serveru a server tak přestane odpovídat. V praxi to znamená, že útočník posílá TCP žádosti o připojení rychleji než cíl útoku zvládá odpovídat. Za normálních okolnosti vypadá handshake takto:

1. klient požádá server o navázaní spojení zprávou SYN,
2. server odešle SYN–ACK zprávu klientovi,
3. klient serveru odpoví ACK zprávou a spojení je navázáno.

V případě útoku, ale útočník posílá opakované zprávy SYN na každý port cílového serveru. Často u toho využívá falešnou IP adresu. Server, který o útoku neví, příjme na první pohled spoustu legitimních žádostí o navázaní spojení a snaží se na každou žádost odpovědět zprávou SYN–ACK. IP adresa, na kterou server zašle odpověď, buď zprávu vůbec neobdrží nebo neodpoví. V obou případech server čeká na odpověď a spojení tak zůstane otevřené, protože server nemůže spojení ukončit. Předtím, než by spojení mělo vypršet, dostane server další SYN zprávu. Proces pokračuje, dokud se nevyčerpají všechny porty nebo server přestane poskytovat služby úplně a legitimní uživatel se tak nemůže připojit.

3.2.3 Záplavové útoky

Útoky jsou zaměřeny na spotřebu šířky pásma sítě a nasycení sítě pomocí botnetů. Botnet je několik počítačů, které spolupracují na jednom stejném úkolu. Při útoku pak uživatelé nemají dostupnost do cílové sítě. Útok se dá jednoduše vytvořit generováním obrovského množství provozu a následným nasměrováním na cílový server. Příklady tohoto typu útoku jsou například UDP a TCP záplavové útoky.

U záplavového útoku UDP [20] útočník přehltí náhodné porty na cílovém zařízení pomocí paketů, které obsahují UDP datagramy. Cílové zařízení příjme všechny pakety, identifikuje s jakou aplikací jsou datagramy spojeny, a protože žádné nenajde, pošle zpět paket „Destination unreachable“. Pakety jsou tak opakovaně posílány a čím víc jich cíl přijme a následně na ně odpoví, tím méně pak zbývá zdrojů na legitimní uživatele a systém je tak přehlcen. UDP je síťový protokol, který nevyžaduje „three – way handshake“ jako protokol TCP. UDP je tedy ideální pro provoz, který vyžaduje menší režii. Stejné vlastnosti však činí UDP zranitelnější. Absence počátečního handshaku na navázaní spojení způsobuje, možnost vyslání velkého množství provozu bez jakékoli ochrany.

[2]synchronizační paket k iniciování handshaku.

3.3 Hping3

Hping3 je síťový nástroj [21], který je schopný odesílat vlastní TCP/IP pakety. Hping zvládá fragmentaci a nastavení paketu téměř na jakoukoliv velikost pomocí příkazové řádky. Hping také podporuje jazyk TCL, který uživateli umožňuje vytvářet vlastní skripty. Skripty následně mohou sloužit například k analýze paketů. Další výhodou nástroje je bezpochyby možnost zobrazení odpovědí od cílových zařízení. V základu je nastaven protokol TCP, ale nástroj podporuje další protokoly jako IP nebo UDP. Hlavičku paketu je opět možno nastavit dle libosti uživatele. Nástroj se používá zejména jako testovací program na síťové zařízení jako jsou firewally, směrovače a další. Autorem nástroje je Salvatore Sanfilippo, který je mimo jiné autorem skenovací techniky idle scan[3], která je důležitou součástí nástroje nmap[4].

3.4 Trafgen

Jedná se o generátor síťového provozu, který je určený k ladění. Využívá paketového rozhraní linuxu, které posouvá kompletní kontrolu nad daty a hlavičkou paketu k uživateli. Využívá výkonný jazyk na konfigurování paketů, který je spíše nízkoúrovňový a není proto omezen na konkretní protokoly. Trafgen [22] může být díky tomu použit pro mnoho účelů s jedinou nevýhodou, která tkví v neschopnosti napodobování reálné relace. Je však velmi užitečný pro testovaní, které má za úkol analyzovat a následně zdokonalit systém, například při Denial of Service útocích.

Ve výchozím nastavení se začíná s maximálním počtem dostupných procesorů. Jakmile uživatel sestaví seznam paketů, které chce používat, může si nastavit počet procesorů, které chce pro tento seznam použít. Nejlepší možný scénář nastává, pokud je přenos uskutečněn z uživatelského prostředí bez jakýchkoliv nepodporovaných modulů nebo modulů třetích stran. V gigabitovém ethernetu má trafgen podobné vlastnosti jako pktgen což je generátor provozu zabudovaný přímo v kernel jádře linuxu. Co se týče konfigurace paketů je trafgen mnohem flexibilnější než pktgen. Trafgen je schopný takzvaného „fuzz“ testovaní, což znamená, že každý nový paket, který je vygenerovaný má jiné hodnoty. Díky vestavěnému protokolu IPv4 může trafgen vyslat ICMP ping na cílové zařízení po každém paketu a otestovat, jestli stále ještě funguje nebo už byl vyřazen z provozu nějakým útokem/testem.

[3]TCP skenovací metoda, která zjistí aktivní služby na cílovém zařízeni.

[4]Network Mapper, neboli nmap je bezplatný, open-source nástroj, která je využívaný převážně na bezpečnostní skenování sítí a hostů.

3.5 Apache bench

Nástroj využívaný pro „load“ a „benchmarking“ testování serveru pro hypertextový protokol (HTTP). Může být spuštěn z příkazového řádku a jeho použití je velmi snadné. Hlavní výhodou nástroje je možnost útok připravit během několika minut s téměř nulovou znalostí o tom, jak útok funguje. Nástroj je vhodný pro úplné začátečníky nebo středně pokročilé uživatele a nevyžaduje žádné složité nastavení. Mnoho uživatelů má tento nástroj k dispozici, protože se na zařízení instaluje automaticky s Apache webovým serverem. Neobsahuje příliš rozšířených funkcí, ale pro základní testovaní je dostačující. [23]

3.6 JMeter

JMeter je součástí projektu Jakarta od Apache [24]. Jedná se o nástroj, který je volně dostupný a vytvořený v programovacím jazyce Java. Slouží k testovaní zátěžových funkcí, chování a k měření výkonu. Původně byl nástroj určený k testovaní webových aplikací, ale rozrostl se i o další testovací funkce.

Nástroj může být použit k testovaní jak statických, tak dynamických prostředků. Může být použit k simulaci velké zátěže serveru, skupiny serverů, sítí nebo objektů, aby se otestovala jejich síla. Mezi další funkce patří analýza celkového výkonu při různých typech zatížení. Mezi stěžejní vlastnosti Apache JMeter patří schopnost testovat webové stránky, Java objekty, TCP, mailové servery a mnoho dalšího. Další výhoda je snadná korelace díky schopností extrahovat data z nejpopulárnějších odpovědí jako je HTML, JSON nebo XML. V neposlední řadě může být jádro rozšířeno pomocí pluginů, například vizuálních.

3.7 Scapy

Nástroj Scapy popsán na oficiálních stránkách [25]. Scapy je naprogramován v Pythonu a umožňuje uživateli odesílat, zkoumat a falšovat síťové pakety. Těchto vlastností pak můžeme využít pro provádění testů. Výhoda nástroje je výkonnost a manipulace s pakety několika protokolů. Zvládá také většinu penetračních úkolů, jako je například skenovaní sítí. Hlavní funkce však zůstává odesílaní a přijímaní paketů. Nástroj umožní útočníkovi vytvořit vlastní set paketů, které pak pošle, přijme na ně odpovědi, porovná je a vrátí výsledek. Hlavní výhoda oproti nástroji hping spočívá v tom, že odpovědí je celý paket.

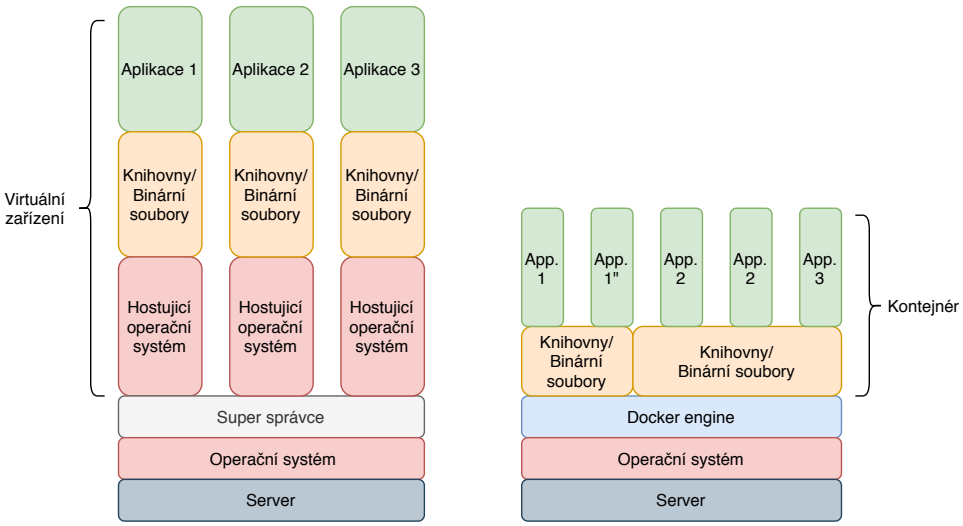
3.8 Grinder

Grinder [26] je Java framework na „load“ testovaní, který usnadňuje spouštění distribuovaného testu. Mezi klíčové výhody patří:

- otestováno je vše, co má rozhraní Java API, to zahrnuje běžné služby jako HTTP, webové servery, webové služby, SOAP a REST, aplikační servery a v neposlední řadě také vlastní protokoly a aplikace.
- Skriptovací testy jsou psány ve výkonných jazycích jako Jython a Clujure.
- Distribuovaná grafická konzole umožňuje sledovat a kontrolovat testy a poskytuje centralizovanou úpravu skriptů.
- Vyspělá HTTP podpora – automatická správa připojení klientů a souborů cookie. Propracovaný záznam interakce mezi prohlížečem a webovou stránkou. Záznam si je možné následně přehrát.

4 Docker

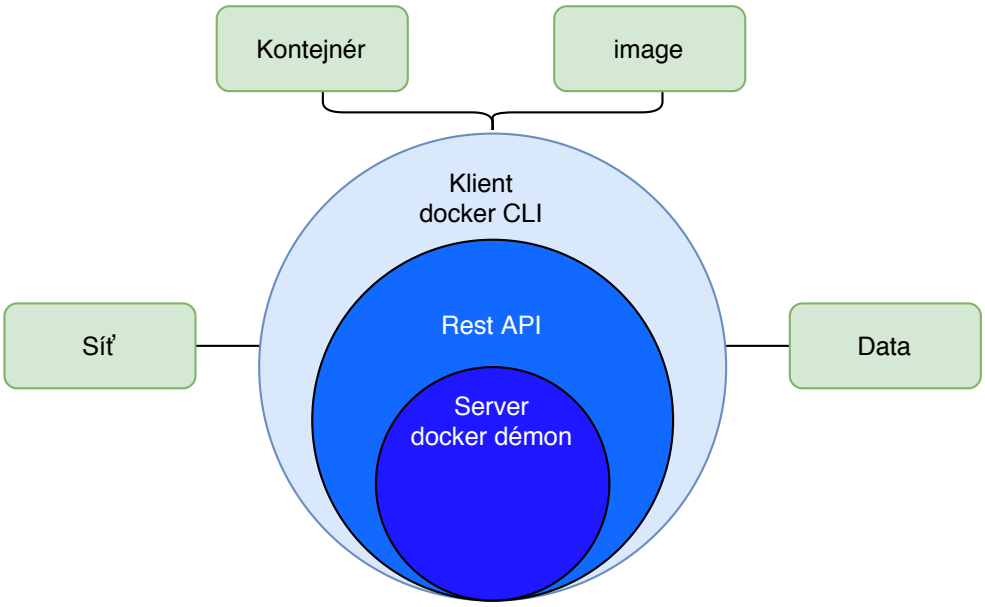
Docker je nástroj určený k usnadnění vytváření, nasazení a spouštění aplikací pomocí kontejnerů. Kontejnery umožňují vývojáři zabalit aplikaci se všemi komponenty, které daná aplikace potřebuje. Jedná se například o knihovny, které jsou pak nasazeny jako jeden balíček. Kontejner vývojáři zajišťuje, že aplikace bude spustitelná na jakémkoli jiném zařízení, které využívá operační systém linux a to bez ohledu na další nastavení, která jsou individuální pro jednotlivé zařízení. Na Docker se dá nahlížet jako na virtuální zařízení, ale s jedním zásadním rozdílem. Virtuální zařízení vytváří celý operační systém, zatímco Docker umožňuje aplikacím využívat stejné kernel jádro, jako systém, na kterém běží. Jediným požadavkem Dockeru je, aby aplikace byly přeposílány s prvky, které již neběží na uživatelově zařízení. Předchozí fakt umožňuje velmi významně zvýšit výkon a zároveň snížit velikost aplikace. V neposlední řadě je pak nutno zmínit, že Docker je open source. To znamená, že každý může Docker rozšířit tak, aby vyhovoval jeho vlastním potřebám. Porovnaní Dockeru s virtualizací lze vidět na obrázku 4.1. [27]



Obr. 4.1: Porovnaní Dockeru a virtualizace

4.1 Nástroje, pojmy a komponenty

Při práci s Dockerem je nutné se seznámit s následujícími pojmy a nástroji. Některé z nich jsou přímo použity v praktické části práce, jiné pak stojí za zmínku z důvodu jejich širokého rozšíření. [29]



Obr. 4.2: Docker Engine

Docker Engine

Jak již bylo zmíněno, Docker je aplikace typu klient–server s těmito hlavními komponenty.

- Server – server Dockeru je démon s názvem dockerd.
- Rest API – rozhraní, díky kterému mohou různé programy komunikovat s démonem a předávat mu tak příkazy.
- Příkazový řádek – slouží k zadávaní příkazů od uživatele.

Všechny výše zmíněné komponenty pak dohromady tvoří Docker Engine, což je znázorněno na obrázku 4.2. [28]

Dockerfile

Při tvorbě každého kontejneru je nutno začít s takzvaným Dockerfilem. Každý kontejner tedy začíná jednoduchým textovým souborem, který obsahuje příkazy a pokyny, kterým Docker rozumí a pomocí nich následně vytváří Docker obraz. V konečné fázi pak Docker Engine spouští příkazy, pokyny a sestavuje obraz. Mezi nejdůležitější příkazy patří:

- FROM – specifikuje, ze kterého počátečního obrazu se bude čerpat,
- COPY – přidává soubory z uživatelova zařízení do obrazu,
- RUN – spouští příkazy v nové vrstvě, používá se pro instalaci softwaru v obrazu,
- CMD – specifikuje, jaké příkazy se mají spustit v kontejneru.

Sandbox

Sandbox označuje prostředí, které je specifické tím, že to, co se stane uvnitř, se nedostane ven. Pokud je proveden příkaz na vymazání uvnitř sandboxu, příkaz je proveden pouze uvnitř, ale hostitelské zařízení zůstane nezměněno. Pokud je například nedopatřením nenávratně poškozena stěžejní část systému, změna se opět nedotkne hostitelského zařízení. Princip sandboxu je v Dockeru široce využíván, více níže.

Docker obraz

Docker obrazy obsahují spustitelný zdrojový kód aplikace a to včetně všech nástrojů a knihoven, které jsou potřeba pro kód aplikace, aby mohla být spuštěná jako kontejner. Jakmile je Docker obraz spuštěn, stává se jednou nebo jednou z mnoha instancí kontejneru. Obraz je možné sestavit kompletně od nuly, což se v praxi většinou neděje, protože velké množství obrazů je volně přístupných a připravených ke stažení z oficiálních, ale i neoficiálních úložišť. Z jednoho základního obrazu je možné vytvořit více Docker obrazů, které sdílí jejich společné rysy v zásobníku. Obrazy jsou tvořeny pomocí vrstev a každá jedna vrstva odpovídá jedné verzi obrazu. Kdykoli se provede změna, vytvoří se nová horní vrstva, která nahradí předchozí vrstvu a stane se aktuální verzí obrazu.

Při každém vytvoření kontejneru z Docker obrazu se vytvoří nová vrstva, která se označuje kontejnerová vrstva. Změny, které jsou pak provedeny v kontejneru (zpravidla se jedná o odstranění, vytvoření nebo upravení souboru), jsou uloženy pouze v kontejnerové vrstvě a existují jen dokud kontejner běží. Tento proces zvětšuje celkovou efektivitu, protože více instancí kontejneru, může běžet z jednoho obrazu a využívat tak společný zásobník.

Docker kontejner

Docker kontejnery jsou běžící instance Docker obrazů. Docker obrazy jsou zpřístupněné jen pro čtení, ale kontejnery jsou běžící a spustitelné. Uživatel může s kontejnerem manipulovat a administrátor pak měnit nastavení a podmínky.

Docker Repository

Docker Repository je internetová nebo síťová služba obsahující Docker obrazy, které jdou následně „pullnout“ nebo „pushnout“ z/do Docker Repository. Uživatel může využít Docker Repository, nebo si vytvořit vlastní. Existují dva druhy úložišť, veřejné (Docker Hub) a soukromé, které je zprostředkováno aplikacemi třetích stran a je často využíváno v rámci korporací.

Docker Hub

Dockere Hub je veřejné úložiště Docker obrazů a lze jej označit jako „největší knihovna pro kontejnerové obrazy na světě“. Obsahuje přes sto tisíc obrazů od komerčních dodavatelů softwaru, open–source projektů a nezávislých vývojářů. Obsahuje i oficiální obrazy vytvořené přímo tvůrci Dockeru. Pro začátečníky se doporučuje využívat certifikované oficiální obrazy. Každý uživatel může na Docker Hub sdílet svůj obraz a taky si jakýkoli stáhnout.

Docker démon

Docker démon běží v hostitelském systému a jedná se o mozek všech operací. Při spuštění kontejneru je zpráva předána démonu, který požadavek vyhodnotí, vybaví s hostitelským operačním systémem a pokud je všechno kladně vyhodnoceno, kontejner je vytvořen. Uživatel s démonem nekomunikuje přímo, ale prostřednictvím Docker klientů.

Docker klient

Docker klient je uživatelské rozhraní pro Docker v binárním formátu. Docker klient přijímá příkazy od uživatele a následně ustanoví komunikaci mezi klientem a démonem. Docker klient pak může komunikovat s jedním nebo více démony.

Docker Swarm

Docker Swarm je interní klastr pro Docker, který umožňuje vytvoření a přístup ke kolekci Docker hostů pomocí Docker nástrojů. Klastr funguje jako API pro Docker a díky tomu všechny nástroje, které komunikují s démonem, mohou využít Swarm pro škálovaní mnoha různých hostů. V současnosti se Swarm už příliš nepoužívá, protože existují vyspělejší klastrové nástroje jako například Kubernetes.

4.2 Architektura Dockeru

Docker využívá architekturu klient–server. Klient komunikuje s démonem, který provádí všechny komplexní operace a následně distribuuje kontejnery. Klient a démon mohou běžet na stejném systému a nebo je možné klienta připojit ke vzdálenému démonu. Klient a démon spolu pak komunikují prostřednictvím Rest API, přes UNIXOVÉ sokety nebo síťové rozhraní.

5 Raspberry Pi

Raspberry Pi jako počítač o velikosti občanského průkazu, který byl navržen a vyroben společností *Raspberry Pi Foundation* [30]. Společnost je nezisková organizace, zaměřená na vytváření počítačů a programovacích instrukcí pro co největší počet lidí. Původním posláním projektu Raspberry Pi bylo dostat levné počítače převážně do rukou studentů, kteří měli zájem učit se programovat. Široká veřejnost však přijala Raspberry Pi velmi pozitivně a rozšířilo se i k programátorům a dalším technologickým nadšencům. Raspberry Pi se začalo používat v projektech, které měly za cíl oživovaní starých arkádových her, ale i pro ovládání robotů. Zařízení je sice malé a od svého zavedení ve svých možnostech velmi pokročilo, ale je důležité zdůraznit, že Raspberry Pi není přímou náhradou za stolní počítač. Nelze na něm spustit tradiční verze Windows, můžeme však spustit celou řadu distribucí operačního systému Linux a to včetně verzí s grafickým rozhraním, webovým prohlížečem a mnoho dalších rozšíření, které bychom hledali u klasického stolního počítače. [31]

5.1 Historie

První Raspberry Pi bylo představené v roce 2012 a dnes se nazývá „Raspberry Pi 1, Model A“. Bylo postavené na mobilním procesoru Broadcom BCM2835, který byl malý, ale poměrně výkonný. Běžně se využíval v mobilních telefonech. Obsahoval CPU, GPU, audio/video processing a další funkcionalitu, to vše na úsporném čipu společně s jednojádrovým ARM procesorem o kmitočtu 700MHz. V následujících letech bylo vydáno několik revizí, které spočívaly například ve výměně čipu Broadcom za vylepšenou verzi. Výměna vedla ke zvýšení výkonu procesoru díky čtyřjádrovému čipu s kmitočtem až 1,2 GHz.

5.2 Deska Raspberry Pi

Pro tuto práci bylo využito Raspberry Pi třetí řady, konkrétně model 3+. Specifikace, které jsou zde uvedeny, pochází z oficiálních webových stránek [32].

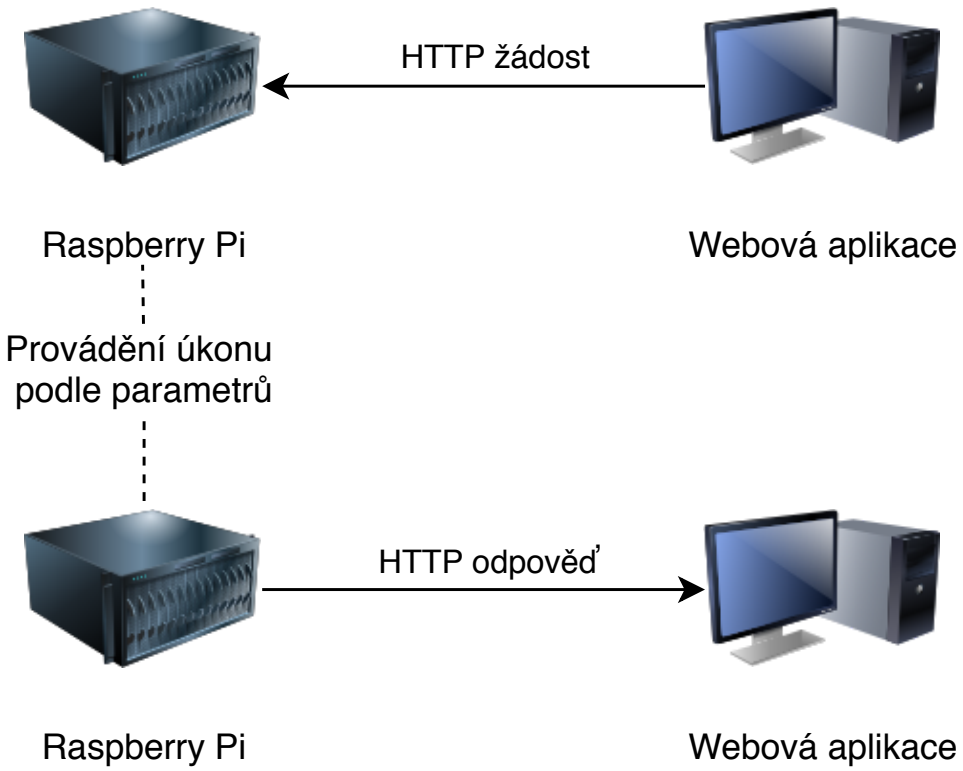
1. Procesor – 1.4 GHz 64-bit quad-core ARM Cortex-A53.
2. Paměť (SDRAM) – 1 GB (sdílená s GPU).
3. Video výstup – HDMI (rev 1.3 a 1.4), 14 HDMI rozlišení od 640×350 do 1920×1200 plus různé PAL a NTSC standardy, kompozitní video (PAL a NTSC), konektor rozhraní displeje (DSI).
4. USB 2.0 porty – 4x.
5. Integrovaná síť – Gigabitový Ethernet skrze USB 2.0 + WiFi 802.11ac, Bluetooth 4.2 BLE a možnost PoE napájení skrze externí HAT modul.

6 Praktická část bakalářské práce

V teoretické části práce bylo cílem seznámit se s protokoly, nástroji a útoky, z nichž nejvhodnější byly vybrány a zasazeny do řešení praktické části. Následující kapitola se věnuje využití konkrétních nástrojů z teoretické části a následnému vytvoření webové aplikace, která komunikuje požadavky se serverem. Server se nachází na zařízení Raspberry Pi, ve kterém běží Docker.

6.1 Zátěžový analyzátor

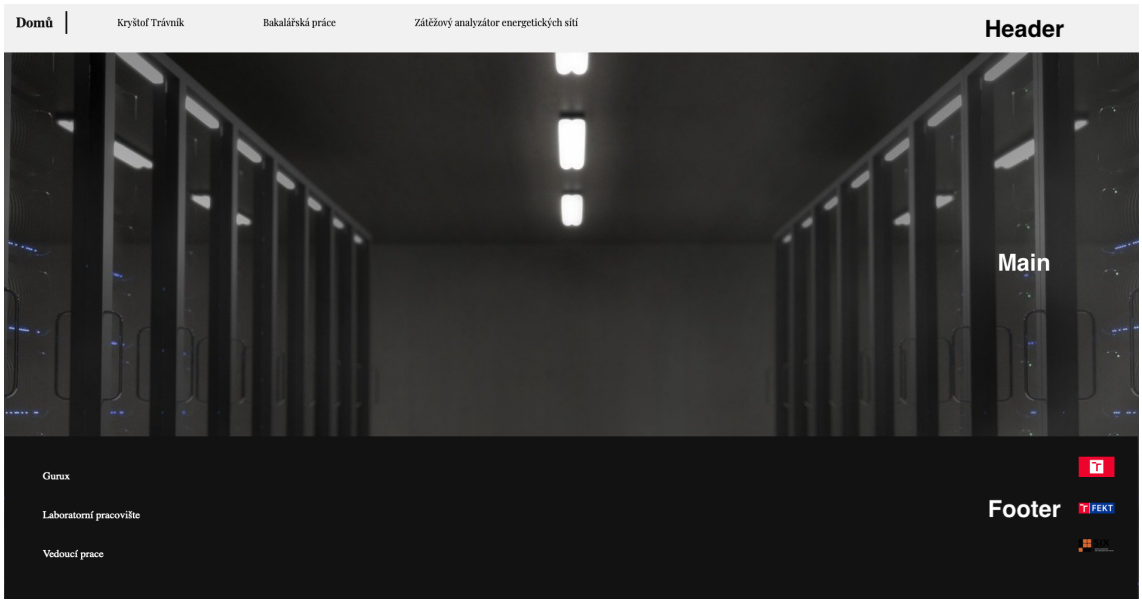
Zátěžový analyzátor je řešení, které se skládá ze dvou hlavních částí. Jedná se o část frontendovou, neboli klientskou, ve které uživatel vykonává veškeré interakce a stanovuje parametry. Druhá část je pak backendová, neboli serverová, kde se naopak odehrávají veškeré úkony na základě stanovených parametrů a po vyřízení jsou výsledky opět zobrazeny ve webové aplikaci a to ve formě grafu. Architekturu komunikace můžeme lze vidět na obrázku 6.1. V serverové části se také nachází veškeré soubory, které jsou nutné pro správné zobrazení klientské části.



Obr. 6.1: Schéma architektury aplikace

6.2 Frontend

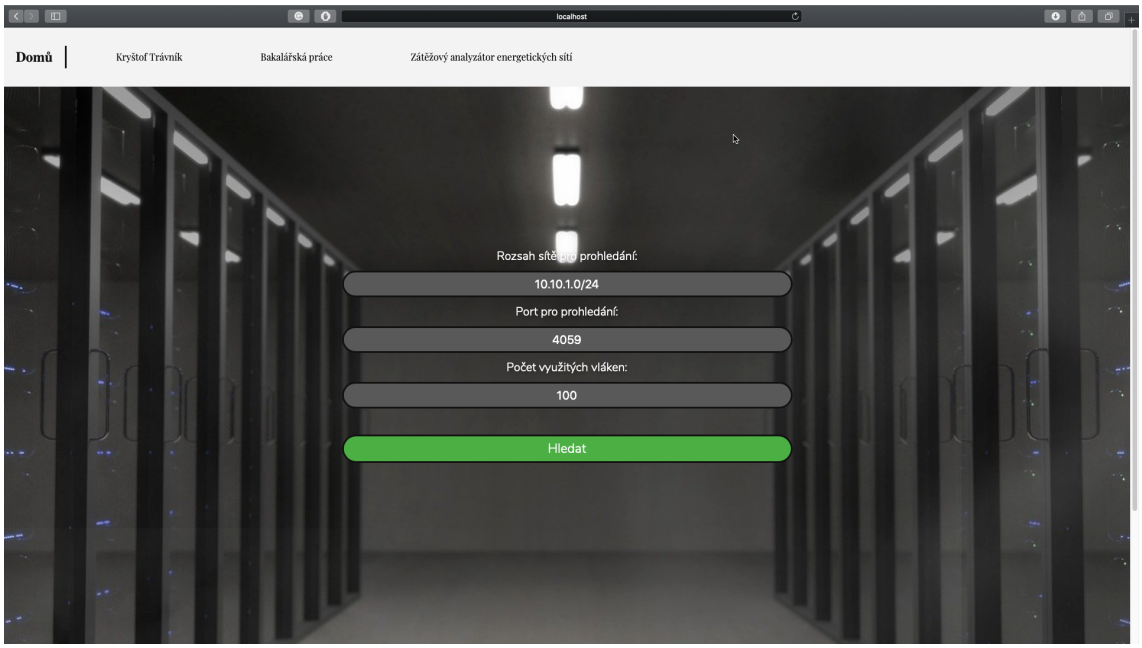
Webová aplikace se ve své klientské části skládá ze 4 PHP stránek, které budou v této kapitole popsány a zobrazeny. Všechny stránky jsou tvořeny převážně z TML kódu, který je graficky upravován pomocí kaskádových stylů (CSS). Menší část pak tvoří PHP kód, který se vykonává na serveru. Výsledná část je prezentována pomocí grafu, který je kompletně vytvořen v JavaScriptu. Každá stránka se skládá ze tří základních částí, které se nazývají Header, Main a Footer. Rozdělení na zmíněné části se používá u většiny moderních stránek. Header a Footer jsou prvky neměnné a nachází se na všech stránkách ve stejné podobě. Header se nachází v horní části stránky (bílé pozadí) a obsahuje tlačítko „Domů“, které nás vždy vrátí na úvodní stránku. Zbytek Headeru obsahuje jméno autora, bakalářskou práci a název programu. Footer se naopak nachází ve spodní části (černé pozadí) a obsahuje hlavně užitečné odkazy ohledně samotné práce, které jsou doplněny o odkazy související s VUT.Část Main se nachází mezi Headerem a Footerem. Jeho součástí je pozadí, které se nemění, ale funkční obsah se na každé stránce liší. Zobrazení stránky lze ilustrovat na obrázku 6.2.



Obr. 6.2: Schéma rozložení stránek

Úvodní stránka

Při zadání adresy stránky je uživatel požádán o přístupové údaje, které jsou ve výchozím nastavení stanoveny jako „admin“ pro uživatelské jméno i heslo. Při úspěšném zadání je uživatel přesměrován na úvodní stránku, index.php, kterou je možno vidět na obrázku 6.3. Úvodní stránka slouží k zadání parametrů pro prohledání sítě

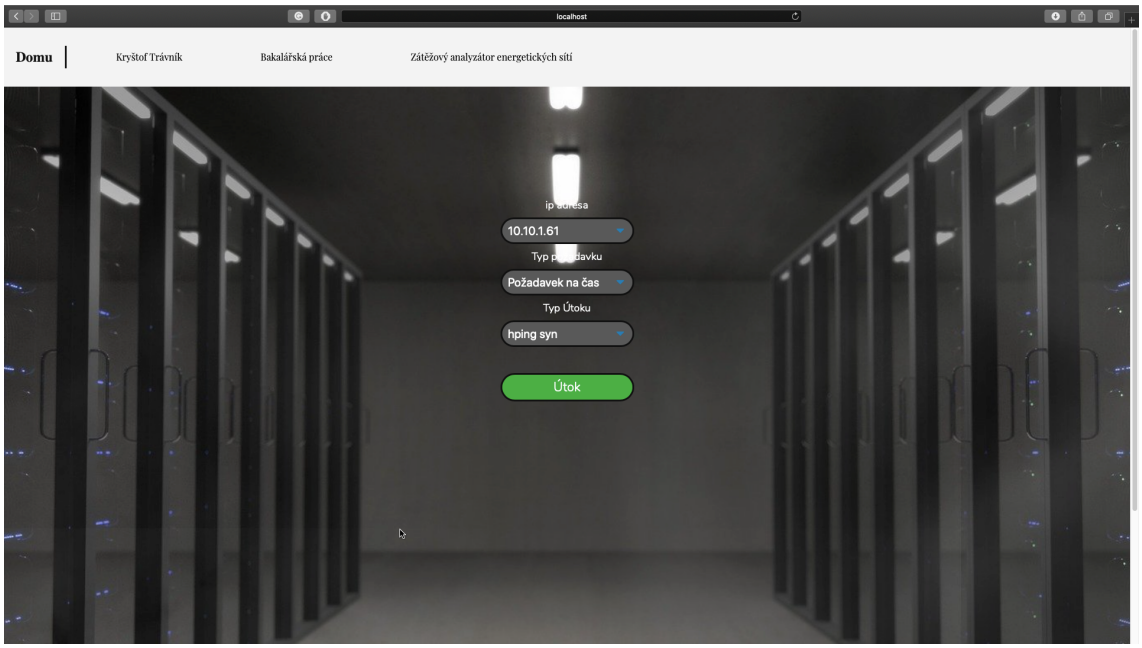


Obr. 6.3: Úvodní stránka

a nalezení smartmetrů. Údaje jsou zadávány pomocí html formuláře a defaultně jsou předvyplněné pro potřebu laboratorního pracoviště. Do prvního pole se zadává IP adresa sítě, kterou chce uživatel prohledat a to ve formátu „IP/maska podsítě“. Toto pole je jako jediné povinné a bez jeho vyplnění nejde postoupit do další fáze. Do druhého pole se pak vyplňuje port, na kterém bude scanner zařízení hledat, a zadává se ve formátu „číslo portu“. Pokud pole není vyplněno, použije se předvyplněné rozmezí portů. Poslední pole slouží k zadání počtu vláken, které má scanner pro prohledání využít. Pole se vyplňuje ve formátu „počet vláken“. Poslední prvek, který se na úvodní stránce vyskytuje, je tlačítko typu submit, nebo-li „Hledat“. Při správném vyplnění a stisknutí tlačítka „Hledat“ je uživatel přesměrován na druhou stránku, „nalezené smartmetry“.

Nalezené smartmetry

Stránka „Nalezené smartmetry“, neboli actionpage.php, kterou lze vidět na obrázku 6.4 se skládá ze tří přepínačů a jednoho tlačítka typu submit pojmenovaného „Útok“. V prvním přepínači jsou prezentovány výsledky ze scanneru z předchozí stránky. Jedná se o IP adresy smartmetrů kde uživatel zvolí cílovou adresu jednoduchým rozkliknutím. Ve druhém přepínači pak uživatel vybírá typ požadavku, který bude na smartmetr poslán. Uživatel má možnost vybrat ze tří různých požadavků. První je čas, druhý průtok elektrické energie a třetí pak napětí. Poslední přepínač je typ útoku. Uživatel v něm vybírá, který útok chce na smartmetr poslat. Na výběr má ze čtyř DOS útoků. První je SYN, druhý ping, třetí FIN a poslední UDP. Následně



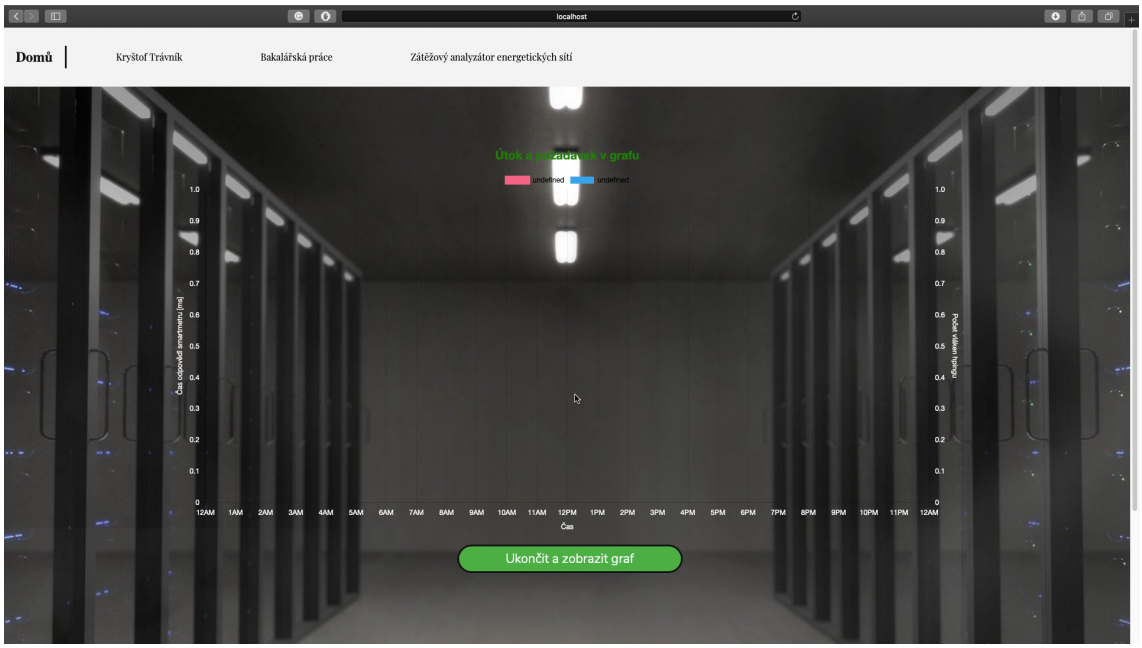
Obr. 6.4: Nalezené smartmetry

uživatel svůj výběr potvrdí stisknutím tlačítka „Útok“ a je přesměrován na stránku „Útok s grafem“.

Útok s grafem

Stránka „Útok s grafem“, neboli actionpage2, je stěžejní pro zobrazení výsledků měření. Hlavní součástí stránky je graf, který je vytvořen z dat v CSV souboru. Graf je vytvořen pomocí JavaScriptu a knihovny Chart.js. Část kódu lze vidět na výpisu 6.1. Na řádku číslo 2 se zvolí, z jakého typu datasetu má být očekáván vstup. Řádek číslo 3 pak přímo určuje, o který soubor se bude jednat. Jedna z nejdůležitějších částí kódu se nachází na řádku číslo 5 až 12. Tato část umožňuje vytvářet graf ze dvou různých datasetů nacházejících se v jednom souboru. Na řádků číslo 13 je pak vytvořen graf a na řádku číslo 17 je stanovena funkce na obnovování hodnot grafu. Následně je pak na řádku číslo 21 nastavena obnovovací frekvence na 3000 milisekund.

```
1 datasource: {
2     type: 'csv',
3     url: 'attack.csv',
4     delimiter: ',',
5     rowMapping: 'datapoint',
6     dataPointLabels: true,
7     datapointLabelMapping: {
8 _dataset: 'dataset',
9 _index: 'time',
```

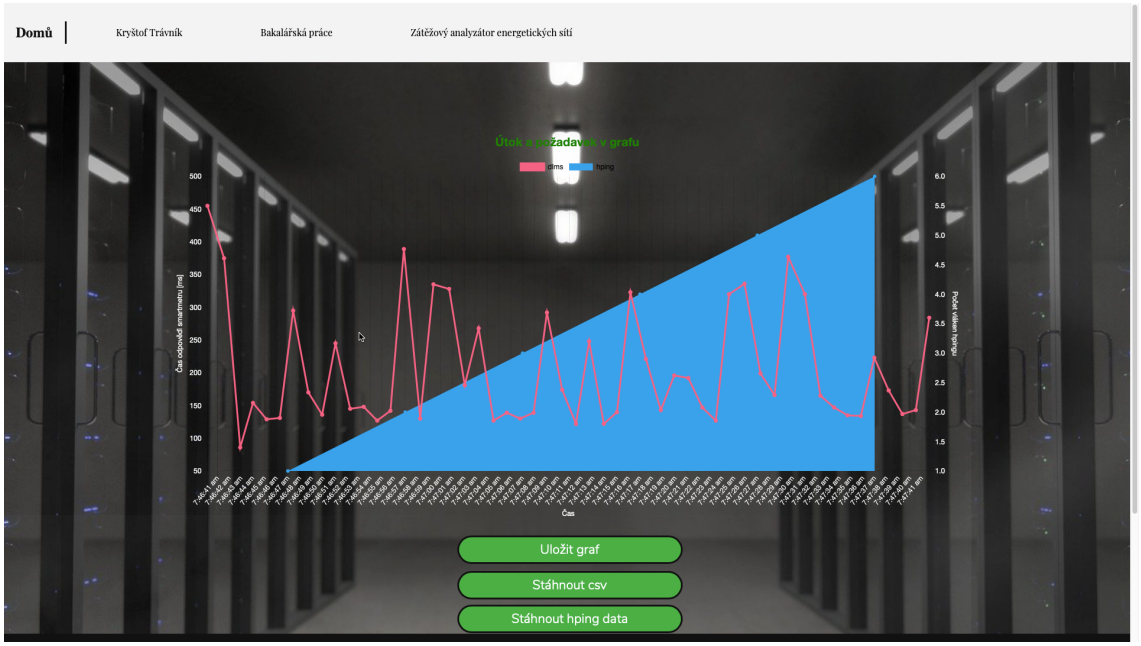


Obr. 6.5: Kostra grafu

```
x: 'time',
y: 'value'
}}}};
function GetData() {
    ctx = document.getElementById('myChart').getContext('2d');
    myGChart = new Chart(ctx, config);
            };
       window.onload = GetData
       function Update() {
       myGChart.update()
            }
    setInterval(Update,3000)
```

Výpis 6.1: JavaScript kód.

Kostra grafu, kterou lze vidět na obrázku 6.5, se vykreslí při prvním načtení stránky a jakmile se vytvoří dataset, graf se začne dynamicky překreslovat. Na levé straně se nachází čas odpovědi smartmetru na zadaný požadavek, čas je udáván v milisekundách. Dolní hodnoty uvádí aktuální čas požadavku a to i z důvodu správného seřazení hodnot. V pravé části je pak údaj o aktuálním počtu vláken nástroje hping3, které jsou spuštěny v daný okamžik. V horní části se nachází ovládáni použitých datasetů, kdy jednoduchým kliknutím můžeme dataset z grafu skrýt a druhým kliknutím pak zobrazit zpátky. Pod grafem je potom tlačítko „Ukončit útok“, které uživatele přesměruje na poslední stránku „Výsledky“.



Obr. 6.6: Výsledná stránka

Výsledky

Na stránce „Výsledky“, neboli cleanup.php se zobrazuje graf v podobě, ve které byl ukončen, což lze vidět na obrázku 6.6. Často nastává menší chyba, která souvisí s obnovovací frekvencí grafu. Při ukončení útoku se zobrazí ještě hodnoty, které se udály po posledním obnovení grafu. Pod grafem jsou pak tři tlačítka. Tlačítko „Uložit“ slouží k uložení aktuálního grafu ve formátu jpg. Tlačítko „Stáhnout CSV“ pak uloží dataset, ze kterého je graf sestaven, a to ve stejném formátu. Poslední tlačítko „Stáhnout hping data“ stáhne CSV soubor, který udává kolik paketů se v jednotlivé fázi hpingu odeslalo.

Použité knihovny a další

V dnešním světě existuje velké množství internetových prohlížečů. V praxi má každý prohlížeč jiné výchozí hodnoty kaskádových stylů. Jelikož jsou styly v této práci kompletně ručně vytvořené (není použit žádný CSS framework jako například Bootstrap) bylo potřeba použít „CSS reset stylesheet“, který zajistil vymazání všech výchozích hodnot, a každý prohlížeč tak zobrazuje nově nastavené styly stejně. Využit byl kód od Richarda Clarka[1]. K nahlédnutí je na začátku souborů style.css a style2.css, které se nachází v přílohách práce.

Ke kaskádovým stylům se dále vážou fonty. Velké množství těchto fontů je za poplatek, ale existují i databáze, které nabízejí fonty zdarma, například „Google

[1] `http://richclarkdesign.com`

fonts“, ze které bylo v práci čerpáno. Využity jsou fonty „Nunito a Playfair“. K nahlédnutí jsou opět ve style.css a style2.css.

K tvorbě grafu byla použita již zmíněna knihovna Chart.js. Knihovna je open source a patří k nejpopulárnějším pro tvorbu standardních grafů. Velké výhody jsou přizpůsobení velikosti stránky, volně dostupná dokumentace, návody a intuitivní automaticky vytvořené tlačítko pro interakci s uživatelem, které již bylo popsáno v popisu jednotlivých stránek. Dalším plusem jsou často předpřipravené komponenty ve smyslu legend a popisků os, které často nevyžadují složité programování jako například v jiné populární knihovně D3.js. Nevýhodou pak může být omezení na nejvíce známe typy grafů a pro komplexnější řešení pak uživatel musí hledat jinde. Poslední nevýhodou je, že graf není možné uložit ve vektorovém formátu. Celá zabalená knihovna zabírá pouze 11Kb, což má malý vliv na výkonnost stránky.

Nakonec jsou pak součásti každé stránky předpřipravené PHP kódy, které zajišťují, že se stránky nebudou ukládat do lokální cache.

6.3 Backend

V backendové částí práce jsou důležité tři prvky.

- Scanner – slouží k nalezení smarmetrů na síti,
- java – navázaní spojení se smartmetrem a následná komunikace,
- skripty – napsané v bashi, které pomocí PHP spojují serverové prvky dohromady a následně pak spojí serverové části s klientskými.

V této podkapitole bude každý jeden prvek rozebrán a to včetně důležitých částí kódu.

6.3.1 Autentizace

Data, která se na stránce nachází, by neměla být volně přístupná, proto se používá autentizace. Stránka je tak chráněna modulem „basic authentication“. Ve složce html se nachází soubor .httacces, který zajišťuje, že přistup na stránku budu povolen jen uživatelům vytvořeným ve souboru passwords. Po úspěšném přihlášení je heslo prohlížečem uloženo a dále pak není po celou dobu vyžadováno. Tento přístup má výhodu v možnosti stahování CSV souboru ze serveru bez opětovného zadání hesla. Nevýhodou může být omezené množství uživatelů, které je možné vytvořit, aniž by se server zatížil. Při chránění citlivějších dat se doporučuje použít modul ssl.

Scanner

Pro potřeby bakalářské práce bylo nutné najít nástroj, který by dokázal identifikovat smartmetery v určitém zadaném rozsahu IP sítě. Jako vhodný pro tento účel se

ukázal programovací jazyk Go[2] a to zejména pro jeho efektivní práci s paralelními procesy (subroutines) a rovněž proto, že je kompilován do binárního kódu (podobně jako jazyk C) a díky tomu je zpracování dané úlohy maximálně rychlé a relativně nenáročné na systémové zdroje. Algoritmus vyhledávání smartmetrů vychází z poměrně jednoduchého předpokladu, že na určité IP adrese běží smartmeter, pokud je otevřen určitý port, ve výchozím nastavení defaultní port 4059. Zda je port otevřen, je zjišťováno pokusem o navázání, což lze vidět ve výpisu 6.2 Pokud daný příkaz

```
conn, err := net.DialTimeout("tcp", fmt.Sprintf("%s:%d", ps.Host,
    ps.Port), ps.Timeout)
```

Výpis 6.2: Zkouška otevřenosti portu.

nevrátí chybu v definovaném maximálním čase (ps.Timeout) a naopak je úspěšně proveden, je port považován za otevřený a IP adresa je prohlášena za vyhledané zařízení. Algoritmus se nachází v souboru scantask.go. Další částí programu je generátor socketů (dvojice IP adresy a portu). V podstatě se jedná o algoritmus, který na základě zadaného rozsahu masky IP sítě, vygeneruje pole socketů, které mají být otestovány, příklad lze vidět na výpisu 6.3. Algoritmus je uložen v souboru socket-

```
func (i Input) getPorts() []int {
  if i.startPort == i.endPort {
    return []int{i.startPort}
  }
  if i.wellKnown {
    return []int{zde jsou všechny well-known ports}
}
  ports := make([]int, i.endPort-i.startPort+1)
  for p := i.startPort; p <= i.endPort; p++ {
    ports[p-i.startPort] = p
  }
  return ports
}
```

Výpis 6.3: Generování socketů.

genrator.go. Scanner byl pojat více obecně, než bylo nezbytně nutné pro potřeby vyhledání smartmeterů, proto je například možné vygenerovat seznam socketů pro tzv. wel-known porty, pokud by bylo třeba zjišťovat zařízení libovolného typu. Poslední částí scaneru je hlavní program (portscanner.go), který jednak zpracovává parametry zadané v příkazové řádce a dále obstarává vytvoření zásobníku paralelních subroutine pro maximální rychlost prohledávání. Základní parametry u scanneru:

[2] `https://golang.org`

- -e (int) – end port (defaultně nastaven port 1024),
- -h (string) – network address (defaultně nastavena adresa "192.168.1.0/24"),
- -l – vypíše pouze aktivní ip adresy,
- -s (int) – start port (defaultně nastaven port 20),
- -t (int) – čas, který má scanner čekat na odpověď (defaultně nastaveno 500),
- -w (int) – počet pracovníků (defaultně nastaveno 100),
- -well – oskenovat pouze známe porty.

Java

Kostra kódu se opírá o již hotové řešení „Generátor DLMS zpráv“, které vypracoval v minulých letech Bc. Kohout [7]. Práce však nesplňovala některé požadavky pro zadání této práce a v některých okruzích ji proto bylo nutné přepracovat. Důležitým aspektem práce je funkčnost v experimentálním laboratorním pracovišti, zřízeném ve školní laboratoři. Úprava vlastní aplikace předpokládala nejdříve pochopení již hotového kódu a následně odstranění nepotřebných částí, což byly převážně části grafického prostředí, a sestavení vlastní třídy, která splní požadavky této praktické části.

V přílohách práce lze nálézt přeložený kód, který obsahuje dva soubory. Jeden z nich je „generatorDLMS“, který byl vytvořen Bc. Kohoutem v minulých letech a obsahuje veškeré potřebné třídy a metody pro navázání spojení. Druhý soubor s názvem „DLMSclient“ je převážně nový kód, který plní požadavky nutné pro tuto práci, ale využívá již hotové metody, které jsou v prvním souboru. Bez prvního souboru by druhý nemohl fungovat. „DLMSclient“ se skládá ze dvou tříd. První třída je NastavKlienta, která je nutná k vytvoření objektu klienta, jež se následně bude k smartmetru připojovat. Kdyby tento klient nebyl vytvořen přesně se stanoveným počtem parametrů, nebylo by možné se k smartmetru připojit, viz přílohy práce [7]. Druhá třída je vytvořena pouze pro účel této práce a nese název App. App obsahuje spustitelnou metodu main. Hned na začátku jsou defaultně nastaveny čtyři proměnné, které lze po převedení Javy do binární podoby libovolně měnit pomocí parametrů příkazové řádky. Možnosti nastavení parametrů lze vidět na výpisu 6.4. Staticky nastavené hodnoty vidíme na řádcích číslo 1 až 4 Uživatel může nastavit libovolný počet argumentů, pokud bude dodrženo pořadí. Program byl vytvořen tak, aby uživatel zadával co nejméně údajů, proto není možné měnit argumenty libovolně. První argument (řádky číslo 5 a 6), který je možné navolit, je IP adresa, v případě této práce to je jedna ze 6 IP adres, kterou našel scaner. Pokud jsou zadány argumenty dva, tak druhým bude nastaven typ požadavku (řádky číslo 8 a 9), který má smartmeter vykonat. V případě zadání tří parametrů bude třetím parametrem číslo (řádky číslo 11 a 12), jež stanovuje kolik požadavků se má na

```
1  String ip ="10.10.1.37";
2  String typPozadavku="time";
3  String pocetOpakovanipozadavku ="1";
4  String timeout = "2000";
5      if (args.length > 0) {
6        ip=args[0];
7      }
8      if (args.length > 1) {
9        typPozadavku=args[1];
10     }
11     if (args.length > 2) {
12       pocetOpakovanipozadavku=args[2];
13     }
14     if (args.length > 3) {
15       timeout=args[3];
16     }
```

Výpis 6.4: Možností parametrů.

zvolený smartmeter poslat. Pokud uživatel zvolí 0, bude počet požadavků nekonečný. Poslední parametr pak v milisekundách (řádky číslo 14 a 15) nastavuje, jaká časová prodleva má mezi dvěma požadavky být.

V další části se pak vytvoří objekt klienta a je navázáno spojení, což lze vidět ve výpisu kódu 6.5. Z důvodu upravení aplikace tak, aby vypisovala hodnoty přímo ve tvaru CSV, si lze všimnout, že veškeré vypsané při úspěšném navázaní jsou zakomentovány. Stejným způsobem je zakomentováno i velké množství hlášek z původního programu.

Poslední důležitá část této třídy je samotné poslání požadavku na smartmetr a změření doby od odeslání do přijetí požadavku. Řešení lze vidět ve výpisu 6.6. Hned na řádku číslo 1 je nastaven začátek měření. Jednoduše se zaznamená reálný čas, kdy byl požadavek odeslán. Na řádku číslo 7 se nachází switch, který podle zadaného parametru zvolí jednu ze tří možností. Na řádcích číslo 9, 14 a 18 se pak pomocí LN jména a OBIS kódu získává požadovaná hodnota. Kromě OBIS se ještě udává číslovka nebo přesněji index, který upřesňuje jakou hodnotu uživatel požaduje. Různé objekty mají jiný počet indexů a pochopitelně i odlišné hodnoty. Na řádku číslo 28 se opět uloží aktuální čas, od kterého se odečte parametr, jež udává časový rozdíl mezi jednotlivými požadavky. V dalším kroku je na řádku číslo 30 vypočítán rozdíl mezi začátkem a koncem požadavku. Poslední důležitá část kódu je na řádku číslo 31, kde se veškeré výsledky vypisují ve formátu CSV, a to v pořadí: dlms, aktuální čas, doba mezi odesláním a přijetím odpovědi.

```
1  NastavKlienta klient = new NastavKlienta(ip, 4059, 17, 1,"
       bakalarka2020", false, false);
2      try {
3        //System.out.println("Connecting to " + ip);
4        klient.getMedia().open();
5        klient.getReader().initializeConnection();
6      } catch (Exception e) {
7        e.printStackTrace();
8        System.exit(1);
9      }
10     if (klient.getReader().Media.isOpen()) {
11       //System.out.println("Connection opened");
12     } else {
13       System.out.println("Not connected");
14     }
```

Výpis 6.5: Vytvoření klienta a navázaní spojení.

Skripty

V poslední podkapitole této části budou rozebrány skripty, které spojují všechny části práce dohromady. Skripty jsou spuštěny pomocí jazyka PHP, příklad jedné PHP operace lze vidět ve výpisu 6.7. První skript, který stojí za zmínku, odesílá útoky pomocí nástroje hping3 a výsledná data ukládá do souboru. Skript, jak lze vidět ve výpisu, se vyskytuje v podobné podobě hned čtyřikrát a to vždy v závislosti od typu útoku, který je odeslán. První typ je SYN a lze ho vidět ve výpise 6.8, a je využit příkaz:

`hping3 -V -c 1000 -d 100 -S -p 4049 --flood IP.`

- -V — pokud by se zobrazovaly odpovědi, hping by je přepsal do čitelnější podoby,
- -c — pokud by se zobrazovaly odpovědi, hping by přestal posílat další pakety po obdržení stanoveného počtu odpovědí,
- -d — nastavuje velikost těla paketu,
- -S — nastaví hodnotu flagu na SYN,
- -p — port destinace,
- —flood — značí že pakety budou posílány co v největším objemu a nebude se čekat na odpověď.

Dalším zvoleným útokem je ping, neboli ICMP flood. Je použit příkaz:

`hping3 -1 --flood -a 192.168.0.1 IP`

S parametry (popsány pouze nezmíněné parametry):

```
Instant start = Instant.now();
         LocalDateTime myObj = LocalDateTime.now();


        try {
          Thread.sleep(b);
          switch (typPozadavku) {
          case "time":
          Object val = reader.read(reader.dlms.getObjects().
              findByLN(ObjectType.CLOCK, "0.0.1.0.0.255"),2);
          reader.showValue(2, val);
          reader.close();
            break;
          case "voltage":
            val = reader.read(reader.dlms.getObjects().findByLN(
                ObjectType.NONE, "1.0.32.5.0.255"),2);
            reader.showValue(2, val);
            reader.close();
          case "power":
            val = reader.read(reader.dlms.getObjects().findByLN(
                ObjectType.NONE, "1.0.31.7.0.255"),2);
            reader.showValue(2, val);
            reader.close();
          default:
            System.out.println("Neznamy typ");
            System.exit(1);
            break;
          }
        }
          catch (java.lang.IllegalStateException | java.lang.
              reflect.InvocationTargetException | java.lang.
              NoClassDefFoundError | java.lang.
              ClassNotFoundException e) {
            Instant end = Instant.now().minusMillis(b);
            Duration timeElapsed = Duration.between(start, end);

            System.out.println("dlms," + myObj.minusNanos(b) + ","
                + timeElapsed.toMillis());

          } catch (Exception e) {
            e.printStackTrace();
          }
```

Výpis 6.6: Záslání požadavku na smartmetr.

```
<?php
        shell_exec('/home/app/allinone.sh syn.sh '.$_GET["metr"].'
           '.$_GET["pozadavek"].'>/dev/null >/dev/null &');
?>
```

Výpis 6.7: Volání skriptu pomocí PHP.

- -1 — budou odeslány Icmp echo žádosti, typ žádosti lze upravit pomocí —icmptype,
- -a — nastavení falešné IP adresy, ze které pakety pochází.

Útok nazvaný tcp je ve skutečnosti typu FIN v tomto znění:

```
hping3 --flood --rand-source -F -p 4049 IP
```

s parametry:

- —rand-source — podobné jako „-a“ s tím rozdílem, že adresa je nastavena u každého paketu libovolně. Využívá se při útocích, která mají za úkol obcházet firewally,
- -F — nastavuje paketu flag s hotnotou FIN.

Poslední útok je typu UDP a je sestaven následovně:

```
hping3 --flood --rand-source --udp -p 4049 IP
```

Jediný nový parametr je —udp, který zasílá UDP pakety na zadané zařízení, defaultně na port 0.

```
#!/bin/bash
h=0
CSVPath="/var/www/html"
while true
do
h=$((h+1))
/usr/sbin/hping3 -V -c 1000 -d 100 -p 4049 --flood $1 &
d1=$(date +"%Y-%m-%dT%T"+02:00)
echo "hping,"${d1}",${h}" >> ${CSVPath}/attack.CSV
sleep 3
done
trap "ps -ef | grep 'hping3' | grep -v grep | awk '{print 2}' |
   xargs -r kill -9" ERR EXIT
```

Výpis 6.8: Jeden z útoků, konkrétně SYN.

Další důležitý skript lze vidět ve výpise 6.9. Slouží ke spuštění souběžného útoku s požadavkem. Skript se spouští se třemi parametry. První z nich je typ útok ve tvaru název.sh. Druhý parametr je IP adresa smartmetru. Třetí je typ požadavku. Druhý i třetí jsou získány přímo pomocí HTTP GET požadavku. První je získán

stejným způsobem, ale vybrán pomocí PHP switche. Ve skriptu se nachází příkazy, které se starají o to, aby veškeré nesprávně ukončené případně i běžící aplikace, byly ukončeny předtím, než je započat nový útok. Skripty jsou často spouštěny na

```
1  #!/bin/bash
2  script=$1
3  ip=$2
4  pozadavek=$3
5
6  ps -ef | grep 'syn.sh' | grep -v allinone.sh | grep -v grep | awk '
       print $2' | xargs -r kill
7  ps -ef | grep 'fin.sh' | grep -v allinone.sh | grep -v grep | awk '
       print $2' | xargs -r kill
8  ps -ef | grep 'ping.sh' | grep -v allinone.sh | grep -v grep | awk
       'print $2' | xargs -r kill
9  ps -ef | grep 'udp.sh' | grep -v allinone.sh | grep -v grep | awk '
       print $2' | xargs -r kill
10 ps -ef | grep dlmsReq.jar | grep -v grep | awk 'print $2' | xargs -
       r kill
11
12 echo "dataset,time,value"> /var/www/html/attack.CSV
13 /usr/bin/java -jar /home/app/dlmsReq.jar ${ip} ${pozadavek} 400
       1000 >> /var/www/html/attack.CSV 2>/dev/null &
14 sleep 10
15 /home/app/${script} ${ip} &
```

Výpis 6.9: Skript pro spuštění útoku i požadavku.

pozadí. Kdyby tak nebylo učiněno, PHP by čekalo na dokončení skriptu a to by způsobilo nekonečné načítání stránky, které by nepřestalo, dokud by nebyl úkon dokončen nebo násilně ukončen přímo na serveru.

Použité knihovny a nástroje

Scanner na vyhledávání smartmerů byl vytvořen v jazyku Golang. Programovací jazyk byl vyvinut společností Google, která potřebovala výkonný programovací jazyk, schopný vykonávat velké množství procesů s relativně nízkou potřebou prostředků. Ne všechny programovací jazyky jsou schopny tak rychle a efektivně řešit problém, jako je nalezení velkého počtu smarmetrů na mohutné a rozsáhlé energetické síti. Jelikož server běží na Raspberry Pi, které je výkonnostně omezené, tak je nižší výkonnostní potřeba prostředků výhodou i zde. Knihovny použité pro scanner jsou součástí Golangu.

V programovacím jazyku Java se využívá knihovna DLMS, která je vytvořena firmou Gurux, spadá pod duální licenci a je open source. Na knihovnu se uplatňuje

licence GNU GPL v2. V práci byly využity dvě tyto knihovny a to gurux.net a gurux.dlms. Obě dvě tyto knihovny pak byly upraveny o kód Bc. Kohouta, které práce využívá taktéž.

6.4 Docker

Celé řešení využívá výhod Dockeru. Z mnoha výhod, které Docker nabízí, je potřeba zdůraznit jednoduchost, se kterou lze celou aplikaci na Raspberry Pi nainstalovat a spustit. Uživateli pouze stačí mít nainstalovaný Docker, mít k dispozici Dockerfile, který je popsán níže, a sestavení. Součástí řešení jsou i skripty a návod, které sestavení ještě ulehčí. Nejdůležitějším prvkem je již zmíněný Dockerfile, který lze vidět ve výpisu 6.10 Na řádku číslo 1 je použito klíčové slovo FROM, které definuje z jakého již hotového Docker obrazu bude čerpáno. K účelu této práce byl zvolen oficiální Docker obraz projektu Apache, který v sobě má již nakonfigurovaný Apache server i PHP. Na řádku číslo 3 je pak argument, který je popsán dále. Řádky číslo 5, 7, 9, a 12 začínají klíčovým slovem RUN, které spustí příkaz, jež následuje. Řádek číslo 5 je nutný pro řešení bugu při instalaci java-openjdk. Řádek číslo 7 pak vytváří uživatele admin, kterému je přiřazeno heslo admin a uloženo do příslušné složky. V případě potřeby je možné přidávat další uživatele, kteří budou mít na stránku přístup. Řádky číslo 9 a 24 jsou důležité pro spouštění operací pomocí PHP, jelikož většina skriptů byla vytvořena uvnitř kontejneru a vše co je vytvořeno uvnitř kontejneru během sestavení, je vytvořeno s právy uživatele root. Dokonce i všechny složky a soubory vytvořené pomocí tohoto Dockerfilu jsou s právy uživatele root. Následně nastával problém při spouštění úkonů při použití PHP, jelikož pokud se spouští příkazy mimo proces webového serveru, nejsou spouštěny s právy uživatele root, nýbrž uživatele www-data, a dochází tak k chybám oprávnění. Řádky číslo 12 až 16 instalují důležité komponenty pro práci jako Javu a hping. Řádky číslo 19 a 20 nastavují potřebné systémové proměnné Javy do cesty, aby bylo možné je spouštět. Řádek 21 umožňuje spouštění hping row paketů uživatelem bez administrátorských práv. Řádek číslo 23 kopíruje soubory nutné pro sestavení webové stránky. Jedná se o obrázky, styly a samotné html kódy. Řádek číslo 25 pak kopíruje skripty, 26 scanner, 27 a 28 binární kód programu v Jave. Na posledním řádku se upřesňuje port, na kterém webový server poběží.

6.5 Pracovní prostředí

Celá práce byla většinu času testována vzdáleně přes nakonfigurovanou VPN. Některé chování při měření v laboratoři nebylo možné plně podchytit. Samotné labo-

```
1  FROM php:7.2-apache
2
3  ARG OS_ARCH
4
5  RUN mkdir -p /usr/share/man/man1
6
7  RUN echo 'admin:$apr1$mUHOIK7P$iMdyQ0haXj6di/SRJHtLZ1' > /etc/
       apache2/passwords
8
9  RUN usermod -u 1000 www-data
10
11 # Install OpenJDK-11
12 RUN apt-get update -y && \
13     apt-get install -y openjdk-11-jdk && \
14     apt-get install -y hping3 && \
15     apt-get install -y libcap2-bin && \
16     apt-get clean;
17
18 # Setup JAVA_HOME -- useful for docker commandline
19 ENV JAVA_HOME /usr/lib/jvm/java-11-openjdk-amd64/
20 RUN export JAVA_HOME
21 RUN setcap cap_net_raw+ep /usr/sbin/hping3
22
23 COPY ./html /var/www/html
24 RUN chown -R www-data /var/www/html
25 COPY ./app /home/app
26 COPY portscanner-${OS_ARCH} /usr/bin/portscanner-linux
27 COPY dlmsReq.jar /usr/bin/dlmsReq.jar
28 COPY dlmsReq.jar /home/app/dlmsReq.jar
29 EXPOSE 80
```

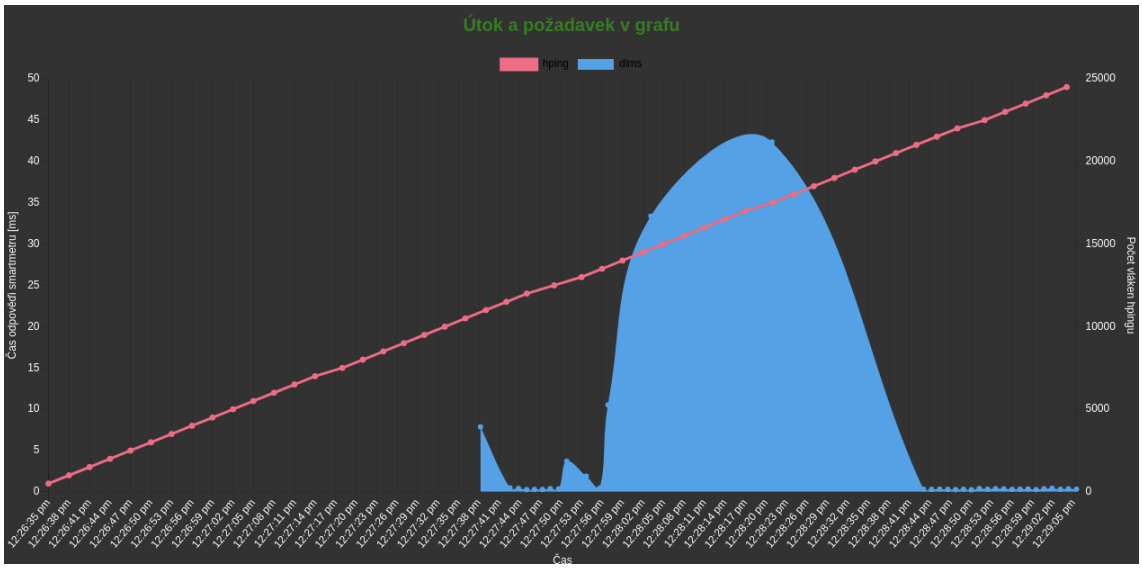
Výpis 6.10: Dockerfile.

ratorní pracoviště se nachází v areálu školy VUT v Brně. Obsahuje standardně šest smartmetrů, které lze dále rozdělit na dvě části. První tři smartmery jsou na IP adresách 10.10.1.37, 10.10.1.61 a 10.10.1.122 a jsou jednofázové, další tři jsou na IP adresách 10.10.1.26, 10.10.1.148 a 10.10.1.158 a jsou třífázové. V síti je dále koncentrátor, který se nachází se na adrese 192.168.88.10. Koncentrátor disponuje webovým rozhraním, na které je možno se připojit při zadaní adresy https://192.168.88.10:8080 a po zadání uživatelského jména i hesla admin, se lze dostat ke statistikám smartmetrů, nebo i možnosti připojení k jednotlivým metrům pomocí telnetu. Veškeré laboratorní prvky, které byly k dispozici ‚jsou chráněny dohodou o mlčenlivosti (důvěrnosti), neboli NDA. Smlouva se uzavírá mezi stranami, které chtějí nějakým způsobem sdílet znalosti, informace nebo data. Jelikož se nepodařilo nalézt řešení, jak rozeznat jednofázový smartmetr od třífázového, uživatel si musí být vědom, jaké požadavky smartmetry zvládají. Vývoj webové aplikace probíhal na monitoru s rozlišením 1920x1080. Aplikace není responzivní. Při jiném rozlišení monitoru se tak může stát, že graf mírně změní polohu ostatních komponent.

6.6 Výsledky

Cílem praktické části bylo měřit odezvu smartmetrů pří paralelním útoku na odepření služeb. Z důvodu nepřístupné laboratoře po většinu akademického roku, bylo veškeré testovaní prováděno vzdáleně. V případě zasílání souběžného útoku na metr, tak dříve než k vyčerpaní kapacity metru, došlo k vyčerpání pásma VPN. Výsledky takového měření nemohly být určeny jako konečné. Zajímavé pak bylo, že průměrný čas první přijaté odpovědi (v DLMS označována jako „getResponseWithDatablock + číslo odpovědi“) při připojení pomocí VPN bylo zhruba 10 sekund, pokud počítáme i dobu navázání spojení. Další odpovědi (getResponseWithDatablock + číslo odpovědi +1 za každou další odpověď) pak přicházely v čase, který byl nastaven jako rozmezí mezi jednotlivými požadavky. Při připojení pomocí nestíněného ethernetového kabelu v laboratoři (bez VPN) se čas první zaznamenané odpovědi zdvojnásobil a dále opět pokračoval bez problémů. Při bližším zkoumáni síťové komunikace bylo zjištěno, že smartmetr odpovídá stále stejně, ale klient nedokáže tuto odpověď přijmout. Chyba pak musí nastávat v Java klientu, konkrétně se ale chybu nepodařilo definovat. Dále pak při připojení v laboratoři nejsou schopné některé metry vždy odpovídat na požadavek. Zpravidla všechny metry dokázaly odpovědět na požadavek času, ale při ostatních pokusech byla odpověď spíše náhodná. Nepodařil se dokázat ani rozdíl mezi jednofázovým a třífázovým metrem. Přes VPN smartmetry odpovídaly na všechny požadavky.

Testovány byly všechny metry a to hlavně pomocí DoS útoků SYN a ping. Webový server běžel v Dockeru na Raspberry Pi. Při útoku SYN bylo průměrně



Obr. 6.7: Výsledný graf

generováno 30 MB/s a při útoku ping pak 5 MB/s. Ani při zvyšovaném počtu paketů odeslaných pomocí hping3 se ve většině případech nepodařilo vyčerpat veškeré prostředky smartmerů a ukončit tak jejich činnost. Důvodem neúspěchu je nejspíše nedostatečný výkon Raspberry Pi v kombinaci s Dockerem. Výhodou řešení práce v Dockeru je možnost propojit více zařízení Raspberry Pi pomocí klastru, například Docker Swarm, a zvyšovat tak výkon, který se bude růst s dalším přidaným Raspberry Pi. Pro ukázku chování smartmetru při větší zátěži bylo použito
PC s procesorem s taktovací frekvencí 2,3GHz a s pamětí 8GB. Proces a výsledný graf lze vidět na obrázku 6.7 V první fázi se smartmetr choval podle očekávání, jelikož dostává pouze útočné pakety s Raspberry Pi. V čase 12:27:53 je spuštěn útok ze zmíněného PC a hodnota času odpovědi se zvedá až na 20 000 milisekund. V čase 12:28:26 je pak útok s PC zastaven a smartmetr se vrací k původnímu času odpovědi 150-300 milisekund. Příklad je uveden pro demonstraci funkčnosti grafu.

Závěr

Cílem bakalářské práce bylo nejdříve v teoretické části nastudovat problematiku DLMS/COSEM protokolu, pochopit princip útoků na vytížení cílového zařízení pomocí DoS útoků a seznámit se s nástroji, které jsou schopny útoky realizovat. Rovněž bylo poskytnuto zařízení Raspberry Pi, které bylo v teoretické části popsáno a v praktické části otestováno. V rámci konzultací pak bylo k práci přidáno řešení využívající Dockeru, které rozšířilo možnosti, kam se případně s prací posouvat dál. Následně byl vytvořen Zátěžový generátor, který je ovládán pomocí interaktivní webové aplikace.

Praktická část se dělí na dvě hlavní části. V první části je popsána klientská strana, ve které se vytváří webové prostředí pro interakci s uživatelem. Ve druhé části se pak popisuje serverová část práce, kde jsou popsány všechny komponenty a jejich fungování dohromady. V serverové části se při řešení naráželo povětšinou na problémy související s právy, které způsoboval především Docker. Bylo proto potřeba detailně nastudovat, jaký uživatel proces spouští a jaká k tomu potřebuje práva. Příklad takového problému je vytvoření CSV souboru manuálně na serveru. Soubor se vytvořil s právy root a při pokusu o zapisování z webového rozhraní byl přístup zamítnut.

Zadání, které bylo stanoveno, bylo splněno, ale výsledky, kterých bylo dosaženo nebyly uspokojivé, ale očekávané. Jednotlivé smartmetry se pomocí různých útoků na odepření služeb uskutečněných pomocí menšího výpočetního výkonu, které nabízelo Raspberry Pi nepodařilo učinit neaktivními a dočasně tak ukončit jejich provoz.

Nejslabší článkem praktické části práce bylo VPN připojení, které bohužel nebylo schopno z důvodu malé šířky pásma přenášet vetší množství dat (klíčové při DoS útoku). Rovněž absence uskutečnění dostatečného množství měření v laboratorním pracovišti neumožnila rozvinutí všech alternativ k úspěšnému vyladění útoku.

Literatura

[1] *Krizový zákon.* In: Sbírka zákonů. Praha: Ministerstvo vnitra, 2000, číslo 240.

[2] *Státní energetická koncepce České republiky* [online]. Praha, 2014 [cit. 2019-12-18]. Dostupné z: <https://www.mpo.cz/assets/dokumenty/52841/60959/636207/priloha006.pdf>.

[3] *Ochrana kritické infrastruktury* [online]. [cit. 2019-12-17]. Dostupné z: <https://bit.ly/2TUM8Kp>.

[4] *Kritická infrastruktura* [online]. [cit. 2019-12-17]. Dostupné z: <https://bit.ly/3dlkjlQ>.

[5] JAGANMOHAN, Reddy, Y. *Industrial Process Automation Systems - Design and Implementation.* Oxford: Elsevier, 2015, s. 359-360. ISBN 978-0-12-800939-0.

[6] *Green Book: Architecture and Protocols* DLMS [online]. Zug, Switzerland: DLMS User Association, 2017 [cit. 2019-12-12]. Dostupné z: <https://www.dlms.com/files/Green-Book-Ed-83-Excerpt.pdf>.

[7] KOHOUT, David. *Zátěžový generátor zpráv DLMS/COSEM.* Brno, 2019. Bakalářská. VUT. Vedoucí práce Ing. Tomáš Lieskovan.

[8] *Blue Book: COSEM Inteface Classes and OBIS Object Identification System.* DLMS [online]. Zug, Switzerland: DLMS User Association, 2017 [cit. 2019-12- 12]. Dostupné z: <https://www.dlms.com/files/Blue-Book-Ed-122-Excerpt.pdf>.

[9] *Yellow Book: Conformance Testing Process.* DLMS [online]. Zug, Switzerland: DLMS User Association, 2017 [cit. 2019-12-12]. Dostupné z: <https://www.dlms.com/files/Yellow-Book-Ed-61-Excerpt.pdf>.

[10] *DLMS FAQ* [online]. DLMS UA [cit. 2019-12-13]. Dostupné z: <https://www.dlms.com/faq>.

[11] *ISO/OSI, IEEE 802.2, and TCP/IP* [online]. 1997 [cit. 2019-12-13]. Dostupné z: <https://bit.ly/3ckPug3>.

[12] *Flag ID List* [online]. DLMS UA [cit. 2019-12-13]. Dostupné z: <https://www.dlms.com/eng/flag-id-list-44143.shtml>.

[13] *International standard IEC 62056-61* [online]. 2002 [cit. 2020-05-26]. Dostupné z: <https://bit.ly/2yNyiSy>.

[14] *Understanding Denial-of-Service Attacks* [online]. 2009 [cit. 2019-12-13]. Dostupné z: <https://www.us-cert.gov/ncas/tips/ST04-015>.

[15] *ICMP ping flood* [online]. 2018 [cit. 2019-12-13]. Dostupné z: <https://www.imperva.com/learn/application-security/ping-icmp-flood/>.

[16] *Denial of Service DDoS attack* [online]. 2018 [cit. 2019-12-13]. Dostupné z: <https://www.geeksforgeeks.org/denial-of-service-ddos-attack/>.

[17] *What Is a DDoS Attack?* [online]. 2019 [cit. 2019-12-13]. Dostupné z: <https://www.thesslstore.com/blog/what-is-a-ddos-attack/>.

[18] *HTTP Flood* [online]. 2018 [cit. 2019-12-13]. Dostupné z: <https://www.imperva.com/learn/application-security/http-flood/>.

[19] *TCP SYN Flood* [online]. 2018 [cit. 2019-12-13]. Dostupné z: <https://www.imperva.com/learn/application-security/syn-flood/>.

[20] *UDP Flood* [online]. 2018 [cit. 2019-12-13]. Dostupné z: <https://www.imperva.com/learn/application-security/udp-flood/>.

[21] *DESCRIPTION of HPING3* [online]. 2012 [cit. 2019-12-13]. Dostupné z: <https://bit.ly/2zeQt3T>.

[22] *DESCRIPTION of TRAFGEN* [online]. 2013 [cit. 2019-12-13]. Dostupné z: <http://man7.org/linux/man-pages/man8/trafgen.8.html>.

[23] *Apache Bench Tutorial* [online]. [cit. 2019-12-13]. Dostupné z: <https://www.tutorialspoint.com/apache_bench/index.htm>.

[24] *Apache JMeter™* [online]. [cit. 2019-12-13]. Dostupné z: <https://jmeter.apache.org>.

[25] *About Scapy* [online]. 2009 [cit. 2019-12-13]. Dostupné z: <https://scapy.readthedocs.io/en/latest/introduction.html#about-scapy>.

[26] *What is The Grinder?* [online]. 2013 [cit. 2019-12-13]. Dostupné z: <http://grinder.sourceforge.net>.

[27] *Docker* [online]. 2020 [cit. 2020-05-20]. Dostupné z: <https://www.ibm.com/cloud/learn/docker>.

[28] *Docker overview* [online]. 2018 [cit. 2020-05-20]. Dostupné z: <https://docs.docker.com/get-started/overview/>.

[29] *Basic Terminologies of Docker* [online]. 2018 [cit. 2020-05-20]. Dostupné z: <https://mindmajix.com/docker/basic-terminologies-of-docker>.

[30] *Raspberrypi* [online]. 2020 [cit. 2020-05-26]. Dostupné z: <https://www.raspberrypi.org>.

[31] *Everything You Need to Know About Getting Started with the Raspberry Pi* [online]. 2017 [cit. 2019-12-13]. Dostupné z: <https://www.howtogeek.com/138281/the-htg-guide-to-getting-started-with-raspberry-pi/>.

[32] *Vlastnosti Raspberry Pi* [online]. 2017 [cit. 2019-12-13]. Dostupné z: <https://bit.ly/3gxk8WG>.

Seznam symbolů, veličin a zkratek

AA	Application Associations
ANSI	American National Standards Institute
APDU	Application Protocol Data Unit
CSS	Cascading Style Sheet
COSEM	Companion Specification for Energy Metering
CSV	Comma Separated Values
DDoS	Distributed Denial of Service
DCS	Distributed control System
DLMS	Device Language Message Specification
DLMS UA	DLMS User Association
DNS	Domain Name System
DoS	Denial of Service
EC-DSA	Elliptic Curve Digital Signature Algorithm
GPU	Graphics Processing Unit
HLS	High Level Security
HTTP	Hypertext Transfer Protocol
HZ	Hertz
ICMP	Internet Control Message Protocol
IP	Internet Protocol
ISO	International Standards Organization
JSON	JavaScript Object Notation
LLS	Low Level Security
NDA	Non-disclosure agreement
MD	Message-Digest
PHP	Hypertext Preprocessor
PLC	Programmable Logic Controller
OBIS	OBject Identification System
OSI	Open Systems Interconnection
SAP	Service Access Point
SCADA	Supervisory Control And Data Acquisition
SHA	Secure Hash Algorithm
SOAP	Simple Object Access Protocol
REST	Representational state transfer
TCP	Transmission Control Protocol
UDP	User Datagram Protokol
URL	Uniform Resource Locator
VPN	Virtual Private Network

XML eXtensible Markup Language

Seznam příloh

A Uživatelská příručka

Příručka popíše, jak nastavit veškeré potřebné komponenty na zařízení v tovarním nastavení, které disponuje Dockerem (Nejlépe zařízení s UNIX operační systémem nebo Raspberry Pi).

1. Otevření v anonymním okně po každém spuštění nejlépe na monitoru s rozlišením 1920x1080.
2. Připojení na VPN nebo připojení přímo v síti, kde se metry nachází.
3. Přepnutí do adresáře */bakalarka*.
4. Spuštění skriptu */build.sh*, který sestaví Docker obraz. Skript nejprve zjistí, jaký procesor, je na daném zařízení použit. Většina instancí operačního systému Linux využívá procesor amd64, ale Raspberry Pi využívá arm. Tento přístup je nutný pro správné fungovaní scanneru. Viz zmíněný argument ARG v Dockerfilu v kapitole 6.4. Podle typu procesoru se pak Docker obraz sestaví se správnou verzí scanneru a příslušnou verzí Apache/PHP.
5. Spuštění skriptu */run1.sh*, který má možnost jednoho parametru (**del**). Pokud je parametr použit, stavající kontejner je zastaven a vymazán. Parametr se využívá především při ladění. Spuštění skriptu bez argumentu sestaví z obrazu Docker kontejner a webový server je tak spuštěn. Nachazí se zde i */run.sh*, který slouží pouze k ladění.
6. Pomocí zadání adresy **localhost:8080/** do webového prohlížeče a vyplnění uživatelského jména **admin** a hesla **admin** je zobrazena úvodní stránka webové aplikace. Další orientace ve webové aplikaci je popsána v kapitole 6.2.
7. Po skončení využívání aplikace se ukončí činnost pomocí příkazu **docker stop <název kontejneru>**.

B Obsah přiloženého CD

Veškerý kód se nachází i na soukromých repositářích, poskytnutých pouze vedoucímu práce. Některé složky nebyly zobrazeny kompletně, ale pouze s popisem, co se v nich nachází.

- `/` ..... kořenový adresář přiloženého CD
  - `app` ..... Obsahuje veškeré skripty
    - `allinone.sh`
    - `cleanup.sh`
    - `syn.sh`
    - `ping.sh`
    - `fin.sh`
    - `udp.sh`
    - `cleanup.sh`
  - `html` ..... Obsahuje veškeré HTML, PHP a CSS komponenty
    - `img.` ..... Obsahuje obrázky a ikony pro webovou stránku
    - `actionpage.php`
    - `actionpage2.php`
    - `cleanup.php`
    - `index.php`
    - `result.php`
    - `info.php`
    - `style.css`
    - `style2.css`
  - `portscanner-linux-amd64` ..... Scanner pro amd64
  - `portscanner-linux-arm` ..... Scanner pro arm
  - `dlmsReq.jar` ..... Java aplikace
  - `build.sh` ..... Docker skript popsán v uživatelské příručce
  - `run.sh` ..... Docker skript popsán v uživatelské příručce
  - `run2.sh` ..... Docker skript popsán v uživatelské příručce
  - `Dockerfile` ..... Dockerfile pro sestavení image
  - `publicjavaApp.zip` ..... Java aplikace
  - `demo.txt` ..... Odkaz na video ukázku
  - `README.txt` ..... Informace